

Xin Yang · Baocai Yin · Xiaopeng Wei

3D Scene Modeling and Robotics Interaction



清华大学出版社
Tsinghua University Press

MOREMEDIA



Springer

3D Scene Modeling and Robotics Interaction

Xin Yang • Baocai Yin • Xiaopeng Wei

3D Scene Modeling and Robotics Interaction



清华大学出版社
Tsinghua University Press



Springer

Xin Yang
School of Computer Science and
Technology
Dalian University of Technology
Dalian, Liaoning, China

Baocai Yin
School of Information Science
and Technology
Beijing University of Technology
Beijing, Beijing, China

Xiaopeng Wei
School of Computer Science and
Technology
Dalian University of Technology
Dalian, Liaoning, China

ISBN 978-981-96-3176-6 ISBN 978-981-96-3177-3 (eBook)
<https://doi.org/10.1007/978-981-96-3177-3>

The original submitted manuscript has been translated into English. The translation was done using artificial intelligence. A subsequent revision was performed by the author(s) to further refine the work and to ensure that the translation is appropriate concerning content and scientific correctness. It may, however, read stylistically different from a conventional translation.

Translation from the Chinese Simplified language edition: “场景建模与机器人感知” by Xin Yang et al., © Tsinghua University Press 2023. Published by Tsinghua University Press. All Rights Reserved.

© Tsinghua University Press 2025

Jointly published with Tsinghua University Press, Beijing, P.R.China.

The print edition is not for sale in Mainland China. Customers from Mainland China please order the print book from: Tsinghua University Press.

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publishers, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publishers nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publishers remain neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

If disposing of this product, please recycle the paper.

*For my readers, who breathe life into
these pages*

Foreword

“AI is probably the most important thing humanity has ever worked on. I think of it as something more profound than electricity or fire.” As Google CEO Sundar Pichai said, intelligence has become the key word in today’s technology development. In the rapid development of intelligent technology, computer image graphics processing technology and computer graphics are like two dancers performing in perfect harmony, reinforcing each other, making a huge contribution to social progress.

Image graphics intelligent processing technology is the deep integration of artificial intelligence research and image graphics processing technology. It is a technology featuring digitization, networking, and intelligence. With the rise of a new technological revolution, image graphics intelligent processing technology has entered a stage of rapid development. With the continuous advancement of computers, artificial intelligence, computer graphics, computer vision, and other technologies, image graphics intelligent processing technology has realized the expansion from single domain to multi-domain, the transformation from single task to multi-task, and the upgrading from traditional algorithm to deep learning.

Image graphics intelligent processing technology is widely used in various industries, having changed the public’s lifestyle and improving work efficiency. Nowadays, image graphics intelligent processing technology has become an important cornerstone in the fields of medicine, autonomous driving, smart security, production manufacturing, gaming entertainment, information security, etc., and it is of great significance to promote industrial technology transformation and optimization and upgrading.

Under the guidance of the “New Generation Artificial Intelligence Development Plan”, artificial intelligence technology is constantly innovating, and the deep integration of artificial intelligence and the real economy has become an important strategic goal. The rapid development of smart cities, intelligent manufacturing, smart healthcare and other fields provides a broad development and application space for the research and application of image graphics intelligent processing

technology. In this context, cultivating professional talents related to image graphics intelligent processing technology for the development of national artificial intelligence has become a demand of the times.

Currently, in the historic intersection of a new round of technological revolution and industrial transformation, image graphics intelligent processing technology is at a critical period. Although image graphics intelligent processing technology has been widely used in many fields, there are still some problems, such as algorithm complexity, data security concerns, model interpretability issues, etc., which also put forward new requirements and challenges for the further research and development of image graphics intelligent processing technology. These challenges come from the continuous updating and iteration of technology, as well as people's constant pursuit and exploration of image graphics intelligent processing technology. How to better improve the visual perception quality of images, how to more accurately extract feature information in images, and how to more scientifically transform, encode, and compress image data have become new directions for domestic and foreign scientific and technological workers and innovative enterprises to explore.

To this end, China Society of Image Graphics and Tsinghua University Press jointly planned the "Image Graphics Intelligent Processing Theory and Technology Frontiers" series of books. The series consists of 21 sub-volumes, covering a wide range of fields and directions with image graphics intelligent processing technology as the main line, and comprehensively introduces the latest research results, technological advances and application practices in this field from the dimensions of intelligent imaging and perception, intelligent image graphics processing technology, intelligent video analyzing technology, 3D vision and virtual reality technology, and visual intelligent application platforms, etc. The purpose of writing this series is to provide the best solution for those who are engaged in the field of intelligent image and graphics processing technology. The purpose of this series is to provide technical references for those who are engaged in the research, development, and application of image and graphics intelligent processing technology, to promote technical exchanges and innovations, and to promote the development and application of image and graphics intelligent processing technology in China. This series will adopt the form of integration of traditional publishing and digital publishing, through the SN-branded ShortURLs into the documents, audio, video, case studies, courseware, and other types of resources to help readers carry out 3D learning and deepen their understanding.

As an important branch of artificial intelligence, image and graphics intelligent processing technology not only needs to constantly innovate core technologies, but also needs to continuously expand application scenarios in various fields, achieving a profound, deep integration of technology and industry. Therefore, at a critical moment when talent is urgently needed, publishing such a series of books is of great significance.

We would like to express our sincere thanks to the authors, reviewers, and Tsinghua University Press for their support and assistance in the preparation of this series. We hope that the publication of this series can provide useful help and

guidance for the readers, promote the development and application of image graphics intelligent processing technology, and push China's image graphics intelligent processing technology to a higher level!

College of Electrical and Information
Engineering, Hunan University
Changsha, China

Yaonan Wang

Preface

General Secretary Xi Jinping pointed out in the report of the 20th National Congress of the Communist Party of China: Education, science and technology, and human resources are the foundational and strategic pillars for building a modern socialist country in all respects. We must regard science and technology as our primary productive force, talent as our primary resource, and innovation as our primary driver of growth. We will fully implement the strategy for invigorating China through science and education, the workforce development strategy, and the innovation-driven development strategy. We will open up new areas and new arenas in development and steadily foster new growth drivers and new strengths. The report also emphasizes: We will promote the integrated and clustered development of strategic emerging industries and cultivate new growth engines such as next-generation information technology, artificial intelligence, biotechnology, new energy, new materials, high-end equipment, and green industry. At present, the overall advancement of the development of new generation artificial intelligence-related disciplines, theoretical modeling, technological innovation, hardware and software upgrades, etc., is triggering chain breakthroughs, accelerating the transition of various fields of economy and society from digitization and networking to intelligence.

General Secretary Xi Jinping pointed out at the Academicians Conference of the Chinese Academy of Sciences and the Academicians Conference of the Chinese Academy of Engineering that robots are the “the pearl on the crown of the manufacturing industry”. Embodied intelligent robots, as carriers of core technology in the field of artificial intelligence, possess the ability to interact with the environment like humans, autonomously plan, decide, act, and execute, playing an important role in serving the country’s major needs, leading the national economic development and ensuring national defense security. Perceiving the 3D environment and understanding the information contained therein is an important manifestation of human intelligence and a prerequisite for free interaction between humans and the environment. Similarly, intelligent robots also need to first fully perceive the environment they are in, which essentially involves 3D digital modeling and understanding of the surrounding scene. The National “14th Five-Year Plan for the Development of the Robot Industry” explicitly states the need to strengthen the cultivation of robotics

talent. At present, although some books involve some of the technologies, there is no reference book that systematically and comprehensively introduces the above fields and connects them into a cohesive whole. This book focuses on the intelligent perception and interaction module in the intelligent robot system, establishes a multidisciplinary cross-integration knowledge system, from the perspective of scene modeling and understanding, discusses the technical frontier and research frontier as comprehensively as possible, and carries out practical discussions on robot navigation, obstacle avoidance, and grasping and other practical application tasks.

This book combines theory and practice, covering both basic algorithms and the latest techniques, with detailed code or pseudo-code resources.

This book consists of 6 chapters. Chapters 1–3 comprehensively introduce the development and application of scene modeling and understanding technology, including 3D geometry and semantic reconstruction, scene active exploration, and scene structure understanding. Chapter 1 introduces 3D geometry and semantic reconstruction algorithms based on traditional multi-view geometry and deep learning, discussing how to recover 3D structure from images. Chapter 2 introduces scene active exploration algorithms in 3D reconstruction and visual semantic navigation, discussing different ways to explore unknown scenes. Chapter 3 introduces two types of scene structure understanding algorithms based on object detection and classification and scene graphs, discussing different ways of scene structure understanding from the two levels of unstructured and structured scene knowledge. Chapters 4 and 5 comprehensively introduce the development and application of robot perception technology, including visual relocalization and robot navigation, obstacle avoidance, grasping, etc. Chapter 4 introduces classic and deep learning-based algorithms related to visual relocalization, navigation, and obstacle avoidance, discussing how robots acquire their own pose status and complete navigation and obstacle avoidance tasks. Chapter 5 introduces analytical methods, empirical methods, and deep learning-based algorithms for robot grasping and interaction, discussing different grasping posture detection and robot grasping methods. Chapter 6 focuses on 3D scene understanding for robot interaction tasks in practice, which is convenient for readers to comprehensively understand and master theory and practice. As for the content of this book, except for Chapter 6, each chapter corresponds to a fixed theme, in which the theory and practice parts are interspersed. In each chapter, the theme is first outlined, the main research work at home and abroad is discussed around this theme, and different technologies are explained according to different methods. The explanation of technology is mainly practical, combined with theoretical knowledge in practice, and “chapter knowledge points” and other characteristic content are designed to facilitate readers to deepen their understanding; each technology includes task description, related work, technical methods, results presentation and summary, etc. The relevant content materials (algorithm code, related papers, etc.) of this book are attached in the form of the SN-branded ShortURLs in each chapter.

Tong Li, Xuefeng Yin, Yuanbo Wang, Yuanyuan Liu, Jiajin Qiu, Hu Lin, Zhaoxuan Zhang, Boyan Wei, Zijian Xu, Xiangji Cheng, Yunduo Zhou, Qirui Bao, Xiaochen Shang, Shanglai Qu, Jianchuan Ding, and others from National and Local

Joint Engineering Laboratory of Computer Aided Design at Dalian University of Technology participated in the partial organization of this manuscript. Some of the research results in this book originate from National 863 Project “Robot Technology as a Networked and Cloud Computing Terminal,” National Natural Science Foundation of China’s key project “Visual and Semantic-based Indoor Scene Understanding and Real-Time Modeling,” Ministry of Science and Technology’s “Brain Science and Brain-like Research” Major Program “Design and Application of Multi-modal and Multi-scale Synergistic Neuroplasticity Brain-like Reinforcement Learning Network Model,” National Natural Science Foundation of China’s Major Research Program cultivation project “Research on Natural Interaction Methods for Co-Integrated Robots for Dynamic Unstructured Environments,” National Natural Science Foundation of China’s General Program “Research on Fast Visual Re-localization Methods for Building Interior Scenes under High-speed and Complex Lighting Environments,” and Dalian Jieqing Program “Intelligent Sensing and Decision Making System for Bionic Robots.”

This book can serve as a teaching reference book in the fields of information and intelligence related majors, computer graphics, computer vision, and intelligent robots in higher education institutions, and can also be used as a reference for professionals in related fields.

In the field of artificial intelligence, the development of scene modeling and robot perception technology is advancing rapidly. Although this book strives to be comprehensive and closely follow its development trend, due to the limited level and time of the authors, there are inevitably omissions in the book. Readers are kindly encouraged to provide feedback and corrections.

Dalian, Liaoning, China
Beijing, China
Dalian, Liaoning, China

Xin Yang
Baocai Yin
Xiaopeng Wei

Contents

- 1 3D Geometry and Semantic Reconstruction 1**
 - 1.1 Overview 1
 - 1.2 Related Work 2
 - 1.2.1 Multi-View 3D Reconstruction 2
 - 1.2.2 Deep Learning-Based 3D Reconstruction 9
 - 1.2.3 Summary 19
 - 1.3 Technical Practice 19
 - 1.3.1 Case Study: Depth Prediction Algorithm Based on Multi-Task Model 19
 - 1.3.2 Case Study: Active 3D Object Reconstruction and Completion Optimization 30
 - 1.3.3 Case Study: Scene Modeling Based on Template Replacement. 42
 - 1.3.4 Case Study: Scene Reconstruction Based on Unsupervised Learning 48
 - 1.3.5 Case Study: Scene Reconstruction Based on Single View Image 53
 - 1.4 Summary 60
 - 1.5 Discussion 60
 - References. 61
- 2 Scene Active Exploration 67**
 - 2.1 Overview 67
 - 2.2 Related Work 68
 - 2.2.1 Scene Exploration in 3D Reconstruction. 68
 - 2.2.2 Scene Exploration in Visual Semantic Navigation 70
 - 2.2.3 Summary 72
 - 2.3 Technology Practice. 73
 - 2.3.1 Case Study: Object-Aware Guidance for Autonomous Scene Reconstruction. 73

2.3.2	Case Study: Design and Implementation of a System for Unknown 3D Scene Reconstruction.	88
2.3.3	Case Study: Semantic Scene Completion-Based Scene Exploration in Visual Semantic Navigation.	108
2.4	Summary	121
2.5	Discussion	122
	References.	123
3	Scene Structure Understanding.	127
3.1	Overview	127
3.2	Related Work	128
3.2.1	Indoor Scene Understanding-Based Object Detection and Classification.	128
3.2.2	Indoor Scene Understanding Based on Discovering Object Associations via Scene Graphs.	130
3.2.3	Summary	130
3.3	Technology Practice.	131
3.3.1	Case Study: Point Cloud Semantic Segmentation and Classification Based on Convolutional Neural Networks	131
3.3.2	Case Study: Scene Graph Generation Based on 3D Scene Point Clouds	148
3.4	Summary	154
3.5	Discussion	154
	References.	155
4	Robot Navigation and Obstacle Avoidance.	157
4.1	Overview	157
4.2	Related Work	159
4.2.1	Classic Relocalization Algorithm	159
4.2.2	Deep Learning-Based Relocalization Algorithm.	168
4.2.3	Classic Navigation and Obstacle Avoidance Algorithms.	170
4.2.4	Deep Learning-Based Navigation and Obstacle Avoidance Algorithm.	171
4.2.5	Summary	174
4.3	Technology Practice.	175
4.3.1	Case Study: Camera Relocalization Based on Feature Database Matching.	175
4.3.2	Case Study: Deep Learning-Based Camera Relocalization	181
4.3.3	Case Study: Robot Navigation Based on Camera Relocalization	193
4.3.4	Case Study: Robot Obstacle Avoidance Based on Known Grid Map	198
4.3.5	Case Study: Multi-Agent Obstacle Avoidance Based on Deep Reinforcement Learning	205

4.3.6	Case Study: Monocular Camera-Based Autonomous Obstacle Avoidance for Complex Scenes	215
4.4	Summary	227
4.5	Discussion	227
	References.....	228
5	Robot Grasping and Interaction	233
5.1	Overview	233
5.2	Related Work	234
5.2.1	Detection of Grasping Methods Based on Analytical Methods and Empirical Methods.....	234
5.2.2	Grasp Detection Based on Deep Learning.....	236
5.2.3	Summary	237
5.3	Technical Practice	238
5.3.1	Case Study: Grasping Pose Detection Based on Object Interaction Dynamics	238
5.3.2	Case Study: Robot Grasping Based on Active Learning ...	249
5.3.3	Case Study: Robot Grasping Based on Faster R-CNN	267
5.3.4	Case Study: Robot Grasping Based on Weakly Supervised Semantic Segmentation Network	281
5.3.5	Case Study: Generative-based Grasping Pose Detection... ..	296
5.4	Summary	303
5.5	Discussion	303
	References.....	304
6	3D Scene Understanding for Robot Interaction Tasks in Practice... ..	309
6.1	Background	309
6.2	Overview	311
6.3	Framework.....	312
6.4	Module Design and Implementation	312
6.4.1	Driving and Information Collection of Multiple RGB-D Cameras	312
6.4.2	3D Reconstruction of Indoor Scenes Based on Multi-RGB-D Camera Fusion.....	323
6.4.3	Scene Labeling Technique Based on RGB-D Sequence ...	341
6.4.4	Platform Design	345
6.5	Summary	355
6.6	Discussion	355
	References.....	356

Abbreviations

ALVINN	Autonomous land vehicle in a neural network
BGNN	Bipartite graph neural network
BN	Batch normalization
BoW	Bag of words
BRIEF	Robust independent elementary features
CNN	Convolutional neural network
DANN	Domain adversarial neural network
DNN	Deep neural network
DoF	Degrees of freedom
DoG	Difference of Gaussian
DQN	Deep Q network
DRL	Deep reinforcement learning
DSAC	Differentiable random sample consensus
DWA	Dynamic window approach
FAST	Features from accelerated segment test
FCN	Fully convolutional networks
GAN	Generative adversarial networks
GAP	Global average pooling
GCNs	Graph convolutional networks
GPU	Graphics processing unit
GRU	Gated recurrent units
IoU	Intersection over union
KD tree	K-dimensional tree
KNN	K-nearest neighbor
LSTM	Long short-term memory
MADA	Multi-adversarial domain adaptation
MDP	Markov decision process
MLP	Multilayer perceptron
MP	Message passing
MRF	Markov random fields
MVCNN	Multi-view convolutional neural network

NBO	Next best object
NBV	Next best view
NetVLAD	Networked local aggregation descriptor vector features
OASC	Object-aware scanning benchmark
OOI	Object of interest
ORB	Oriented fast and rotated brief
ORCA	Optimal reciprocal collision avoidance method
PCA	Principal component analysis
PLY	Polygon file format
PPO	Proximal policy optimization
RANSAC	Random sample consensus algorithm
ReLU	Rectified linear unit
RL	Reinforcement learning
RMSE	Root mean square error
RNN	Recurrent neural network
ROS	Robot operating system
SAD	Sum of absolute differences
SAVN	Self-adaptive visual navigation method
SCARF	Supervised classification applied to road following
SFM	Structure from motion
SIFT	Scale invariant feature transform
SLAM	Simultaneous localization and mapping
SPL	Success weighted by path length
SSD	Sum of squared differences
STN	Spatial transformer networks
SURF	Speeded up robust features
TF-IDF	Term frequency-inverse document frequency
TSDF	Truncated signed distance function
URDF	Unified robot description format
URLs	Uniform resource locators
VAAL	Variational adversarial active learning strategy
VLV	Learning value from videos
VSF	Viewing score field
WILDCAT	Weakly supervised semantic segmentation network
YARF	Yet another road follower

Chapter 1

3D Geometry and Semantic Reconstruction



1.1 Overview

In the realms of computer graphics and computer vision, we commonly utilize images to describe objects or scenes. However, in structural representation, 3D models offer a more intricate portrayal of local intricacies and formations compared to 2D images. A comprehensive 3D model can circumvent issues like self-occlusion and structural gaps in 2D images, thereby enhancing the precision of other visual tasks such as object recognition and segmentation. It is evident that, in comparison to images, 3D models possess robust expressive capabilities and hold an indispensable role in domains such as robot navigation, interaction, and data visualization.

The essence of 3D reconstruction lies in crafting a 3D model for a scene or object. The prevalent approach involves leveraging readily obtainable images and amalgamating images captured from diverse viewpoints into a unified global 3D coordinate system to reconstruct its 3D form. This chapter predominantly delves into the process of extracting 3D structure from images, encompassing depth information estimation, 3D object reconstruction, and 3D scene reconstruction and elucidates fundamental principles alongside showcasing pertinent practical outcomes.

Conventional methods of image-based 3D reconstruction typically entail employing a multitude of images with known camera positions to formulate a 3D geometric representation of the scene. This geometric representation may manifest as a disparity map (or depth map), a 3D volume comprising voxels, a distance field, or a 3D point cloud, among other forms. The fundamental tenet of 3D reconstruction is straightforward, ensuring that the generated 3D points align with the corresponding input images from the camera positions when projected back onto the 2D image. This underscores two pivotal factors in 3D reconstruction—namely, the localization of image observation points and the correspondence between observation points and the reconstruction model, also known as localization and mapping. In scenarios where camera positions are not calibrated during the capture process, it becomes imperative to ascertain camera localization to establish the conversion relationship

between pixel points and the 3D model, ultimately culminating in the construction of a 3D map within a global coordinate framework.

Building upon the aforementioned foundational principles, 3D reconstruction can leverage the alignment between visual data and geometric attributes for matching, thereby facilitating pose optimization and 3D information retrieval, a concept termed multi-view 3D reconstruction in this chapter. Alternatively, the burgeoning approach of deep learning-based 3D reconstruction transforms the problem into a neural network training paradigm, integrating projection geometric relationships within an end-to-end network to bolster the network's adherence to geometric constraints.

Nevertheless, compared to 2D images, 3D models necessitate greater storage capacity and computational resources. Coupled with the inherent complexity of 3D models, the reconstruction and analysis of 3D structures pose heightened challenges. The two methods outlined above exhibit limitations in certain aspects. This chapter expounds on the fundamental principles of each method, elucidating the domestic and international research landscape, dissecting their strengths, weaknesses, and suitable scenarios, while also presenting various technical cases pertinent to 3D reconstruction (including depth information estimation, 3D object modeling, and 3D scene reconstruction), ultimately concluding with a comprehensive summary of the chapter's content.

► sn.pub/d18778

The foundational algorithms and technical cases discussed in this chapter can be explored through the links provided in the SN-branded shortURLs. The SN-branded shortURLs can be scanned by the SN More Media app. These resources are publicly accessible on GitHub, enabling readers to deepen their comprehension by delving into relevant papers and code repositories.

1.2 Related Work

1.2.1 Multi-View 3D Reconstruction

Multi-view 3D reconstruction methods utilize a large number of images with known camera positions to generate a 3D geometric representation of the scene. This geometric representation can be a disparity map (or depth map), a 3D body composed of voxels, a distance field, or a 3D point cloud. The basic principle is that the output 3D points need to be consistent with the corresponding view image when projected back to the 2D image, that is, according to the projection association between the image and the 3D global structure, the 3D geometric structure is recovered from a set of images. For example, the stereo vision method reconstructs the surface from a single-camera video in real time [1, 2], this method assumes that all surfaces are ideal scattering, and the depth of points in the scene is determined by minimizing the energy function, so the reconstruction result is less robust. With the advent of

consumer-grade depth cameras [3], 3D reconstruction methods that integrate RGB-D information (RGB represents color images, D represents depth maps) are becoming more and more popular. Compared with single-view reconstruction, depth information can help determine the z-coordinate of image points in the camera coordinate system, enhancing the robustness of reconstruction in some environments. For example, RGB-D SLAM [4, 5] using RGB-D images extracts SURF (Speeded Up Robust Features) feature points from the images, matches feature points between frames to estimate the camera pose of RGB-D frames, and finally uses graph optimization methods to optimize the pose trajectory, achieving a globally optimal reconstruction result.

The KinectFusion algorithms [6, 7] only use depth information, iteratively update the truncated signed distance function (TSDF), and finally feed back to the user through 3D rendering. Compared with the aforementioned methods, the KinectFusion algorithm greatly reduces the computational load during the scene creation process, basically meeting the real-time requirements of user interaction, but due to memory limitations, it can only reconstruct in a small range of scenes. Based on this limitation, some teams have improved by using spatial stratification [8] and hash voxels [9], or by using point clouds [10, 11], a more generalized data form, to increase resolution.

In recent years, more representative 3D reconstruction works include the DynamicFusion algorithm [12], which solves the problem of how to perform real-time surface reconstruction when the object being measured is moving, based on the KinectFusion algorithm; the BundleFusion algorithm [13] is currently the best method in this field, it does not require any explicit loop detection and supports real-time robust tracking on the graphics processing unit (GPU). The ROSEFusion algorithm [14] from our National University of Defense Technology is suitable for scene reconstruction by fast-moving cameras.

This chapter briefly describes the principles of the classic visual simultaneous localization and mapping (SLAM) and KinectFusion algorithms. It is believed that after understanding these, the reader will be able to understand and master other more advanced algorithms by reading related literature.

1.2.1.1 Visual SLAM Algorithm

SLAM refers to the process of driving a machine or handheld sensor without prior knowledge of the environment, moving and sensing information in the environment, and establishing a map model of this unknown environment while estimating its own motion. The basic process of the visual SLAM algorithm will be introduced from the mathematical description of SLAM and the classic visual SLAM framework.

In terms of location, if the sensor's trajectory is separated into corresponding position sequences according to discrete time steps $t = 1, \dots, K$, and x represents the position of the robot or other agent, then this position sequence can be represented as $x_1, \dots, x_K$; In terms of mapping, if it is assumed that the map is composed of N

landmarks (represented by $y_1, \dots, y_N$), landmarks can be seen as some distinctive building points in the plane map, and as some 3D points in the 3D scene.

As the sensor moves along the trajectory, a portion of the landmark points will be measured at each moment, and their observation data will be obtained. The SLAM problem has become a study of the change in the sensor's position on the trajectory and the measurement of landmarks, the former is called motion, and the latter is called observation. The SLAM model is an algorithm that represents and estimates the states by mathematically formulating the motion and observation components.

Suppose that at $t = k$, the value observed by the moving sensor at position x_k is u_k , and the sensor noise is w_k , then the motion equation is defined as:

$$x_k = f(x_{k-1}, u_k, w_k) \quad (1.1)$$

The motion equation represents the change in position compared to the previous moment. Suppose that a landmark point y_j is observed at position x_k , and the observation data $z_{k,j}$ is obtained, then the observation equation is defined as:

$$z_{k,j} = f(y_j, x_k, v_{k,j}) \quad (1.2)$$

where $v_{k,j}$ is the noise of the observation. The motion equations and observation equations mathematically describe the process of simultaneous localization and SLAM, and localization and mapping are to solve the positions x and landmark points y in these two equations, so the essence of the SLAM problem becomes a state estimation problem of solving two equations, that is, estimating the position and landmark point variables in the equation through noisy observation data.

There are various methods to solve this state estimation problem, including linear Gaussian systems, Kalman filters, particle filters, and graph optimization methods commonly used in visual SLAM. Since the main purpose of this chapter is to apply 3D reconstruction methods, the following mainly elaborates on visual SLAM technology using visual information.

Visual SLAM mainly uses visual sensors (such as monocular cameras, binocular cameras, and depth cameras) to collect data and locate itself, while building a map of the scene. Since most of the data collected is in the form of images, some computer vision methods can be used to assist in the state estimation of the positioning equation and the observation equation.

The classic visual SLAM can be divided into five modules, which are information reading, visual odometer, loop detection, backend optimization, and mapping. The relationship between the five modules is shown in Fig. 1.1.

1. **Information Reading:** This module mainly reads and processes the image information obtained by the sensor; its role is to provide necessary visual information for establishing a visual odometer. In actual operation of the robot, it may not be limited to reading visual information, it may also include reading and synchronizing inertial sensor information.

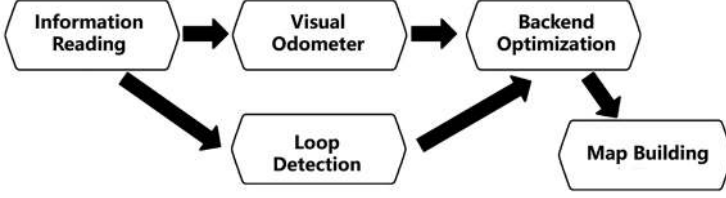


Fig. 1.1 Relationship of the five modules of SLAM

2. **Visual Odometer:** The main function of this module is to estimate the movement of the camera based on the read visual information and build a map according to the movement of the camera. Although there are errors in the map built, it will be optimized at the back end. To complete a visual odometer, feature point method or direct method can be used. Taking the feature point method as an example, first, extract the features of the image and match the features between adjacent frames, then use the epipolar geometry method or the nearest point iteration method to solve the pose change of the camera based on the matched features, and finally use the transformation formula between the homogeneous coordinates under the world coordinate system and the pixel coordinate system to convert the 2D feature points into landmark points on the 3D map. The conversion relationship is as follows (1.3):

$$P_{uv} = KTP_w \quad (1.3)$$

where K is the intrinsic matrix of the camera, which is fixed after the camera is manufactured; T is the extrinsic matrix of the camera, which represents the rotation and translation of the camera, and changes with the change of the camera position.

After obtaining the position changes of adjacent frames, the position of each image in the entire sequence is cumulatively calculated, and the 2D feature points are converted into 3D landmark points to obtain a mileage map.

3. **Loop Detection:** A visual odometer can build a map, but due to the presence of noise, there are errors in the pose estimation between adjacent frames. After a mileage, the errors accumulate, causing the map rebuilt when the camera returns to the previous location to be different from the original. Therefore, detecting the previously visited location and forming a loop can add constraints on the loop on the basis of feature matching between adjacent frames, and the position of the camera and the construction of the map should satisfy the global consistency. Regarding loop detection, a common method is to build a Bag of Words (BoW) model for the image. This model describes what kinds of features are on the image. For example, if there are tables, chairs, and books on the image, then these objects constitute the “words” in BoW. Many words form a dictionary through clustering. If a word exists in the image, it is represented by 1, if not, it is represented by 0. Then if there are N words in the dictionary, an image can be represented by an N -dimensional vector composed of 0 and 1. Calculating the

distance between vectors can judge the similarity between two images. When the similarity is high enough, it can be considered that a loop has been detected, thus providing more constraints for backend optimization.

4. **Backend Optimization:** Backend optimization receives the camera position calculated by the visual odometer and the constraints provided by loop detection, and finally obtains a globally consistent trajectory and map through optimization. There are many methods for backend optimization, including linear Gaussian systems, Kalman filters, particle filters, and graph optimization methods. Taking the commonly used graph optimization method as an example, first project the points in the world coordinate system to the pixel coordinates according to formula (1.3) to obtain the observation equation of visual SLAM and abstract it as

$$z = h(x, y) \quad (1.4)$$

where position x refers to the camera pose, that is, the rotation matrix and translation vector, suppose its corresponding Lie algebra is ξ . When the map to be reconstructed is a 3D point cloud, the landmark y refers to the 3D point p , and z refers to the value obtained by the observation equation. Then the observation error can be defined as the difference between the sensor observation value and the observation equation estimated value:

$$e = z - h(\xi, p) \quad (1.5)$$

Suppose there are m poses and n landmarks, then the overall cost function can be defined by the least squares method:

$$L = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n |\xi_{ij} p_{ij}|^2 \quad (1.6)$$

Here, the subscript ij represents the data of the i th position and the j th landmark. Next, we need to optimize this error. By solving with the least squares method, it is equivalent to adjusting the hidden states (poses and landmarks) in the equation, and finally obtaining a globally consistent pose trajectory and map.

5. **Map Building:** There are many forms of map representation, including 2D grid maps, topological maps, point cloud maps, mesh maps, and octree maps. Taking the 3D point cloud map as an example, it is represented by a set of 3D coordinate points, the data is unordered, and it accurately represents the specific spatial location of objects in the map. It is divided into sparse maps and dense maps. Sparse maps only select some representative parts to form the map, equivalent to landmarks, and others are ignored. Dense maps model everything that the sensor can see, and maps used for navigation usually convert it into many small cubes, or voxels, according to a certain resolution. The voxel represents whether there is a 3D structure occupying it.

After these five modules, the system can obtain image information through visual sensors and perform real-time positioning and map building, but the actual application may be far more complicated than this, and the information used may not be limited to visual information, which is not considered in this chapter.

1.2.1.2 KinectFusion Algorithm

KinectFusion [6, 7] algorithm is a 3D modeling algorithm that uses depth information captured by depth sensors to track camera trajectories and reconstruct scenes in real time. Compared with visual SLAM methods, KinectFusion algorithm can be divided into four stages: The first stage is the conversion of depth maps, where depth information converts 2D pixels to 3D coordinates; the second stage uses the iterative closest point method to calculate the change in camera pose; the third stage is equivalent to mapping in SLAM, but in the form of 3D data, KinectFusion algorithm no longer uses point clouds but distance fields; and the fourth stage is real-time 3D rendering for users. The following introduces these four stages.

1. **Depth Map Conversion:** After the depth camera obtains the depth map, the pixel points can be converted into a 3D point in the camera coordinate system through the calibrated camera intrinsic matrix K . The conversion formula is

$$V_{x,y} = D_{x,y} K^{-1} [x,y,1]^T \quad (1.7)$$

Here, $D_{x,y}$ is the depth value of the depth map at the image coordinate (x, y) , and $V_{x,y}$ is the corresponding 3D point in the camera coordinate system. Considering the neighboring projection points, the vector of the 3D point can also be calculated:

$$n_{x,y} = (V_{x+1,y} - V_{x,y}) \times (-V_{x,y}) \quad (1.8)$$

Points in different camera coordinate systems need to be uniformly converted to a global world coordinate system, which requires calculating the rotation and translation of the image corresponding to the camera, that is, the camera's extrinsic matrix T . The calculation of the extrinsic matrix is carried out in the next stage of camera tracking.

2. **Camera Tracking:** The KinectFusion algorithm uses the Iterative Closest Point method for real-time calculation of camera pose. The algorithm first uses projection association to project vertices V in the current camera coordinate system to its camera coordinate system according to the camera pose calculated at the previous moment, and then projects them into image coordinates, looking for the position of the corresponding points in the image according to the vertices and normals. Next, after eliminating outliers and judging whether the distance angle between the vertex and the corresponding point in the global coordinate system is within a certain threshold, a set of corresponding points is obtained. Finally, in

order to obtain the optimal extrinsic matrix T globally, the distance between each point on the current frame and the tangent plane of the corresponding point on the previous frame is calculated. The optimal extrinsic matrix T is obtained by iteratively minimizing this distance. The KinectFusion algorithm linearly approximates this system by assuming that there is only one incremental transformation between frames and solves it on the GPU through Tree Reduction. Throughout the process, the KinectFusion algorithm solves on the GPU, without extracting features or sparse sampling of points, and has good real-time performance. However, the premise is that it requires dense camera tracking, that is, it requires a dense sequence of depth maps to assume that there is only one incremental transformation between frames.

3. **Volumetric Representation:** For the camera pose trajectory calculated by tracking, the KinectFusion algorithm [6, 7] uses volumetric representation to integrate the depth information on the trajectory and unify it to a global coordinate system. This volumetric representation is actually a distance field, representing its distance to the actual surface. When it is in front of the surface, the value is positive, when it is behind the surface, it is negative, and when it is on the surface, it is 0. If only the vicinity of the surface is considered, a truncation area can be defined, and the distance function under the truncation area is called the truncated signed distance function (TSDF). If a scene is represented by 512^3 voxels, each voxel contains a TSDF value. The KinectFusion algorithm can update the TSDF value in real time. The updated distance field can be further rendered in 3D to present to the user.
4. **3D Rendering:** After obtaining the updated distance field, the KinectFusion algorithm uses ray tracing to render and track the implicit surface in the 3D volumetric representation, intuitively presenting it to the user. For pixels in the rendered image, the GPU first renders in parallel along the ray, and the ray will pass through the voxels in the 3D body. Next, on the voxels on both sides of the TSDF value of 0 on the ray, the intersection point of the implicit surface is calculated by the linear interpolation method of three sampling points, and the surface normal is calculated by deriving at the zero intersection. Finally, the calculated information is used for pixel illumination calculation to complete the rendering.

After these four stages, the system can track the camera's trajectory in real time under the GPU through the depth information obtained by the depth camera, establish a global volumetric representation, which is equivalent to a 3D mapping, and finally can be rendered to the user.

These 3D reconstruction algorithms based on multi-view geometry have become increasingly mature after many years of research, but these algorithms initially extract feature points, match feature points between frames, and optimize the pose, which inevitably faces the following problems:

First, the algorithm requires the information perceived by the sensor to be stable and not affected by the environment. Whether using a color camera or a depth camera, the collected data will be affected by the environment. For color images, because it is necessary to extract its features, it requires the image to have enough

texture, but the texture of scenes like walls in the photo is not rich, making feature matching difficult. At the same time, the quality of color images and depth images of strong reflective objects and transparent objects will be affected. In addition, instability and disturbances during movement may cause frame loss during scanning or make the shot pictures blurry, all of which will affect the accuracy of matching, thereby affecting positioning and mapping.

Second, the algorithm requires dense viewpoints. Both the SLAM algorithm's matching of adjacent frames and the iterative closest point method in the KinectFusion algorithm require dense shooting angles, so that there is enough overlap area for matching and pose estimation. This is different from human intelligence, as humans can often determine the 3D structure of an object by observing from a few visual angles. Using such dense visual angles to scan an object in this era of artificial intelligence seems somewhat "wasteful." Especially in some scenarios, due to obstructions or inaccessibility, some viewpoints of the object cannot be obtained, and the reconstruction may lose key structures at this time.

Third, the algorithm requires manual path planning. Both methods scan the scene or object from dense viewpoints, requiring dense image information covering the scene or object. There is no algorithmic scanning trajectory planning, which requires manual scanning with a camera to ensure enough information is covered.

Despite these limitations, these methods are still the most popular 3D reconstruction algorithms due to their relatively mature algorithms, good real-time performance, and robustness and accuracy that meet the requirements in general environments. They will have stronger applicability with the update of hardware and algorithms.

1.2.2 Deep Learning-Based 3D Reconstruction

As mentioned earlier, 3D reconstruction methods based on multi-view geometry have many limitations, including the need for dense viewpoints, the assumption that there are no mirror reflections in the scene, the need for rich textures, and the need to optimize camera registration errors. These limitations lead to the possibility of missing key structures in the reconstruction results in environments with poor lighting, obstructed shooting angles, and strong noise. In contrast, with the support of a large amount of data, deep learning can predict the global structure of objects based on prior knowledge and complete local details. The introduction of deep learning brings new ideas to the research of 3D modeling and provides new possibilities for 3D modeling of sparse information, making it a hot topic in academia and industry.

If the 3D reconstruction problem is seen as a prediction and generation problem based on image features, a 2D encoding network and a 3D decoding network can be built using convolutional neural networks, thereby encoding 2D pixels into feature vectors and decoding them into 3D voxels. For example, Wu et al. [15] use a self-encoding 3D adversarial generation network, Rohit et al. [16] use a T-L network, which can generate corresponding 3D models from a single input image. Dai et al.

[17] combine the self-encoding network structure with the classification network, and through post-processing of graphic synthesis technology, input a single view-point image, output the predicted and synthesized 3D model. The Princeton team uses a dilated convolution network to complete the scene and semantic segmentation from a single viewpoint depth map [18].

In addition to using 3D voxels to represent the reconstructed shape of an object, the object can also be reconstructed by generating octree voxels, point clouds, 3D meshes, and implicit. For example, the octree generation network [19] uses more efficient octrees to reduce memory usage and computation, generating higher resolution 3D voxels. The point cloud generation network [20] is the first to introduce the more convenient data form of unordered point clouds into deep learning, predicting the distribution of its 3D points from a single-color image. The AtalsNet algorithm, based on the generation of point clouds, judges the surface of the point and reconstructs a 3D mesh with a surface [21]. The DeepSDF algorithm and the OccNet algorithm use implicit representations to depict 3D shapes, outputting distance functions or decision boundaries in continuous 3D space, and generating 3D meshes through sampling and isosurface extraction [22, 23].

A common point of the above methods is that they input a single-view color image or depth image, train neural networks based on large 3D model databases (such as ShapeNet [24], SUN3D [25], and ScanNet [26]), and predict 3D information through the idea of “2D encoding–3D decoding.” However, even if a powerful deep learning model is trained, it cannot compensate for the lack of view, especially for complex 3D models. Obtaining only one view makes it difficult to express all geometric structure information, resulting in potentially significant errors in the final reconstruction. To address this issue, Choy et al. [27] proposed a 3D recurrent neural network that receives images taken from multiple views and uses the hidden layers of the network to implicitly represent the current geometric structure being reconstructed, thereby establishing a deep learning-based 3D reconstruction framework that combines single and multiple views. Kar et al. [28] further improved the voxel reconstruction quality by projecting semantic features from different views into the corresponding 3D space to extract geometric features of images, based on this network model. The team from the Dalian University of Technology combined multi-view 3D reconstruction and view planning, dynamically selected views using a 3D attention mechanism, and recovered 3D information through deep learning [29].

There are many other 3D reconstruction algorithms that use deep learning technology, such as template matching methods that achieve high-quality reconstruction by matching parts in the database and assembling them [30, 31], and methods that achieve 3D reconstruction with primitive representation through primitive fitting [32, 33]. The author may not be able to provide a detailed introduction to all relevant methods but will start from the most basic algorithms in deep learning-based 3D reconstruction methods, explain their processes and pros and cons, and help readers have a basic understanding. It is believed that with this understanding, readers will be able to read the latest related literature more easily.

1.2.2.1 Brief Description of Deep Learning Methods

Deep learning has been a very popular research field in the past 10 years. In April 2013, the MIT Technology Review listed deep learning as the top breakthrough technology of 2013. After AlphaGo defeated the world champion of Go, deep learning once again set off a craze and was applied in many areas such as medical, entertainment, and transportation. Deep learning mimics the working process of the human brain, receiving and processing signals, continuously abstracting these signals, and finally completing tasks. The human perception process is from shallow to deep, and deep learning also uses this feature to abstract low-level features into high-level features (such as the category of an object) from shallow to deep. Formally, it is like a neural network, processing input signals through neurons, and abstracting features for deeper cognition as the network deepens.

In computer vision, convolutional neural networks and recurrent neural networks are the two most popular deep learning models. The former is used to recognize displacement, scaling, and other forms of distortion invariance of 2D pixels or 3D voxels, while the latter deals with time series problems where the current output is related to the previous output.

1. **Convolutional Neural Network:** A typical convolutional neural network mainly consists of an input layer, convolutional layer, activation function, pooling layer, and fully connected layer. The computation of the convolutional layer involves two key operations: establishing filters and sliding windows.

As shown in Fig. 1.2, taking the 2D convolution of an image as an example, the input layer processes the input image and sends it to the convolution layer. The role of the convolution layer is to perform deeper feature extraction on the input. In the convolution calculation, the convolution kernel, or filter, must first be set. The size of the filter for 2D convolution is $fH \times fW \times inCh \times outCh$, where fH and fW represent the height and width of the convolution kernel, and $inCh$ and $outCh$ represent the depth (number of channels) of the input and output features. On each channel of the input and output, there is a pixel window of

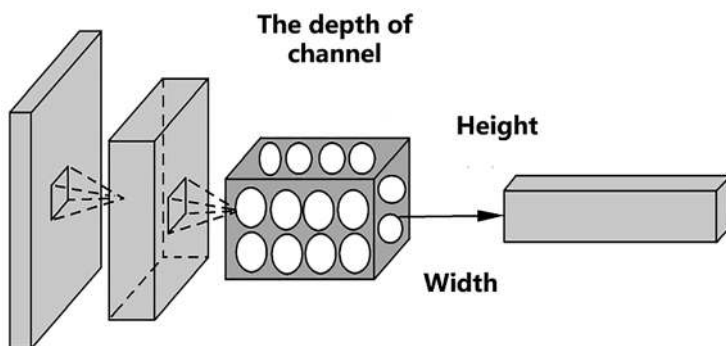


Fig. 1.2 Convolutional neural network logical structure

$fH \times fW$, under which are the pixel values of the input features. The weighted sum of the weights and pixel values under the calculation window is the feature value corresponding to the center position of the window on the output channel. By sliding the window and calculating the weighted sum under the window, all pixel values of this output feature can be obtained. Of course, this is not accurate. There will be no window center coverage at the feature edge, but it can be achieved by padding values to output the entire complete feature.

There are various forms of activation functions according to different needs, such as Sigmoid activation function, Relu activation function, and Tanh activation function. The characteristics and usage of different functions will not be elaborated here.

The role of convolution operation is to deepen the features, while the role of the pooling layer is to compress the feature map. Similar to convolution, the pooling operation also has a sliding window, but the pooling operation does not have weights, but downsamples the pixels in the window, such as taking the average or maximum value, and downsamples the feature map in the window to a pixel. As the number of layers in the convolutional neural network increases, the length and width of the feature map gradually decrease, and the depth gradually increases.

The essence of the fully connected layer is vector multiplication. Let x be the input feature vector of the fully connected layer, and W and b represent the weight and offset of the fully connected layer, then the output feature vector y is

$$y = Wx + b \quad (1.9)$$

The fully connected layer plays the role of a “classifier” in the convolutional neural network, and “compresses” the semantic information of the original input features. There features can be used for different tasks such as classification and regression.

In summary, the convolutional neural network uses a deep network structure to perform local perception of the feature map, then integrates local information at a higher level, and finally obtains global information, which is in line with the characteristics of local information reception by human visual cortex neurons. The weight-sharing feature of the convolution kernel makes it have fewer parameters than the full connection and a smaller computational load.

2. **Recurrent Neural Network:** The output of the convolutional neural network only considers the current input, but many times the input of the problem to be solved is a sequence, such as video streams and text. And the result at each moment in the sequence may not only be affected by the current moment’s input but also by the input of previous moments. At this time, the convolutional neural network that is only affected by the current moment’s input is no longer competent, and the recurrent neural network (RNN) that specifically handles time series problems can be used.

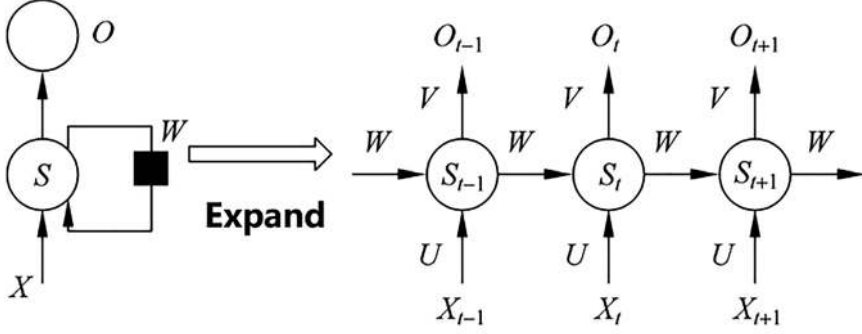


Fig. 1.3 RNN logical structure

The recurrent neural network saves the previous information at each time step through the hidden layer and uses it for the current moment's calculation. In terms of structure, the nodes between the hidden layers are connected, and the input of the hidden layer is not only the current input but also the output of the previous moment.

Figure 1.3 intuitively shows the logical structure of the RNN. The network is composed of an input layer, a hidden layer, and an output layer. As can be seen from the figure, the hidden layer retains the output of the previous moment, and together with the current input, it is input into the hidden layer, and finally the output of the network is obtained. Its mathematical expression can be seen in eq. (1.10):

$$\begin{aligned} S_t &= f(W S_{t-1} + U X_t) \\ O_t &= g(V S_t) \end{aligned} \quad (1.10)$$

wherein, S_t is the hidden layer state at time t , X_t , and O_t are the input and output at time t , respectively, f and g are activation functions, and W , U , and V are the weights in the network. The weights of the recurrent neural network share parameters at each moment.

However, such a structure has some problems. When the time series is long, gradient vanishing or gradient explosion may occur during backpropagation. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) can solve this problem well.

LSTM (the logical structure is shown in Fig. 1.4a) adds four gates to the ordinary RNN to control the removal or addition of information to the hidden layer of the network. These four gates are the forget gate that decides what information to discard, the input gate and cell gate that decide what information to store, and the output gate that updates the old cell state in the hidden layer. The mathematical representations of the four gates are

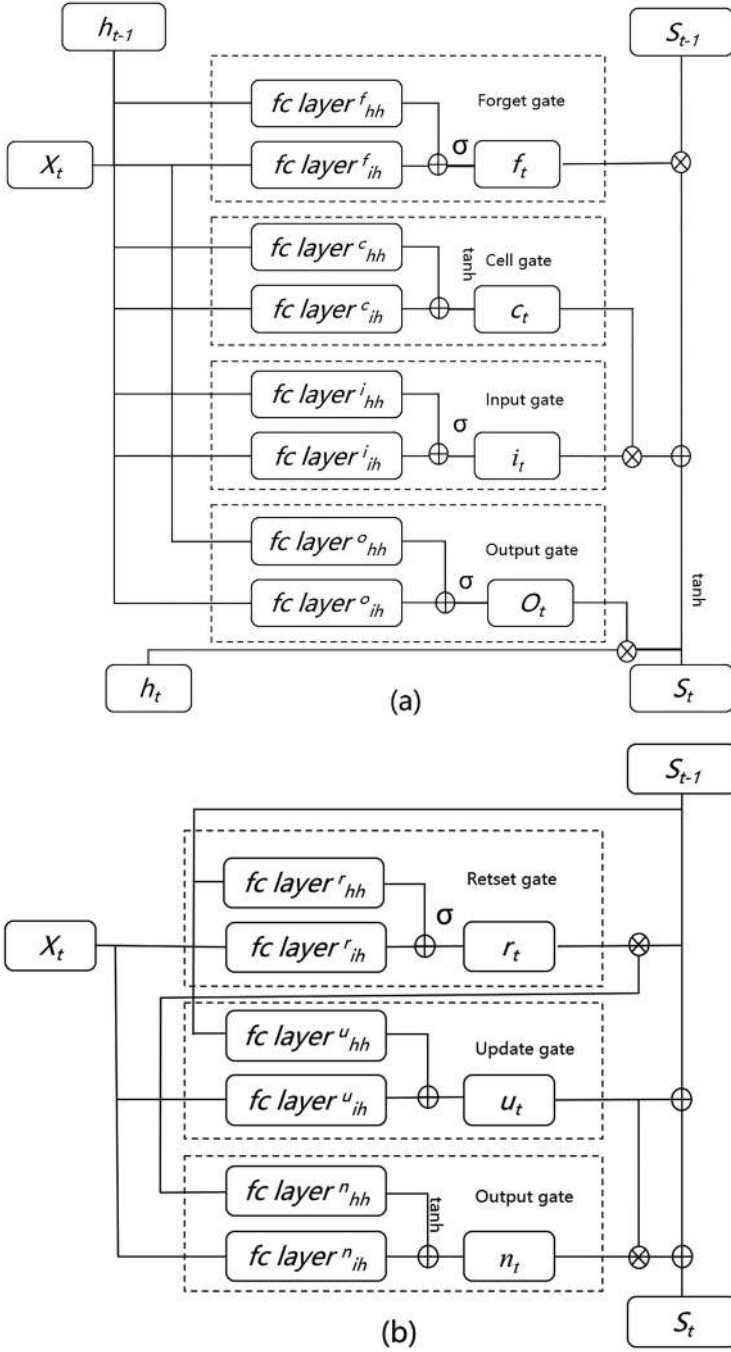


Fig. 1.4 LSTM (a) and GRU (b) logical structure

$$\begin{aligned}
g_f &= \text{sigmoid}(W_{i,f}x_t + W_{h,f}h_{t-1} + b_f) \\
g_{in} &= \text{sigmoid}(W_{i,in}x_t + W_{h,in}h_{t-1} + b_{in}) \\
g_c &= \tanh(W_{i,c}x_t + W_{h,c}h_{t-1} + b_c) \\
g_o &= \text{sigmoid}(W_{i,o}x_t + W_{h,o}h_{t-1} + b_o)
\end{aligned} \tag{1.11}$$

where the subscripts f , in , c , and o represent the forget gate, input gate, cell gate, and output gate, respectively. W_i and W_h represent the weights from the input to the gate and from the hidden layer to the gate, respectively, and b represents the bias. With the control of the four gates over the information, we can get a hidden layer state and output that can selectively remember and forget:

$$\begin{aligned}
c_t &= g_f * c_{t-1} + g_{in} * g_c \\
h_t &= g_o * \tan(c_t)
\end{aligned} \tag{1.12}$$

where c_t is the cell state at time t ; h_t is the hidden layer state at time t , which is also the final output. From LSTM, it can be seen that it can selectively retain and forget information to determine the update and output of the hidden layer. However, it has more parameters and a more complex model. GRU, as a variant of LSTM, has fewer parameters and a simpler structure and is a popular recurrent neural network.

GRU combines the forget gate and input gate into an update gate and also merges the cell state and hidden state. Its structure is shown in Fig. 1.4b, and the mathematical expression is as follows (1.13):

$$\begin{aligned}
g_r &= \text{sigmoid}(W_{i,r}x_t + W_{h,r}h_{t-1} + b_r) \\
g_u &= \text{sigmoid}(W_{i,u}x_t + W_{h,u}h_{t-1} + b_u) \\
h_t &= (1 - g_u) * \tanh(W_{c,i}x_t + W_{h,c}h_{t-1} + b_n) + g_u * h_{t-1}
\end{aligned} \tag{1.13}$$

where the subscripts r , u , and n represent the reset gate, update gate, and output layer, respectively. Tests have shown that both GRU and LSTM can ensure that important features are not lost in long-term propagation and effectively solve the problems of gradient explosion and gradient vanishing during back-propagation. In numerical comparison, the two are not much different, both are better than ordinary RNN. But GRU has fewer parameters than LSTM, smaller computational load; therefore, it can save memory and time.

1.2.2.2 Reconstruction Based on 3D Neural Network

Understanding the 3D world through images is a very popular research problem in the field of computer vision. People can take photos of the 3D world with a camera and can also imagine its 3D shape based on the 2D image of the object. Similarly, given a 2D image of an object from a single viewpoint, can we output its 3D shape?

This is unimaginable for traditional 3D image reconstruction methods because a single image cannot calculate the position of the camera or the depth value of the pixels, and even if this information is given, the structure of the occluded part will be missing due to the lack of information from a single viewpoint.

A feasible method is to provide prior knowledge about the shape of the 3D object and use this prior knowledge to predict the parts that are not observed from a single viewpoint. The method of deep learning is based on this idea, by building a neural network to establish a mapping from 2D images to 3D shapes, and using a large dataset of images and 3D shapes for training, ultimately achieving an end-to-end network that inputs single-view images and outputs corresponding 3D shapes. Convolutional neural networks, as a network commonly used to extract 2D and 3D features, are often used in single-view reconstruction.

A simple single-view reconstruction network structure is shown in Fig. 1.5. It adopts a 2D encoding and 3D decoding structure. The 2D encoding consists of multiple 2D convolutional layers and fully connected layers, which serve to extract the 2D feature map of the input image. As the network layers deepen, the depth of its feature map also deepens, and finally, a one-dimensional feature vector is obtained under the fully connected layer.

This feature vector compresses and expresses the semantic features of the input image, equivalent to encoding the input in terms of features. Let $h(\cdot)$ represents the 2D encoding network, and I represents the input image, then the encoding feature $h(I)$ can be obtained.

The 3D decoding consists of multiple 3D deconvolution layers. Deconvolution is the backward propagation of convolution, a reverse operation compared to convolution, and the downsampling in convolution is also replaced with upsampling in deconvolution. Therefore, the deconvolution network continuously enlarges the features, the depth becomes shallower and shallower, and it gets closer and closer to the shape of the input or output data, which is equivalent to decoding the encoding feature $h(I)$. In terms of convolution operations, 3D convolution is similar to 2D convolution, except that the convolution kernel has become 3D, and the features being processed have changed from 2D pixels to 3D voxels.

A 3D voxel refers to a grid composed of cubic voxel grids. The value stored in each voxel indicates whether there is a 3D shape occupying this small grid. The voxel occupancy probability is between 0 and 1. The higher the voxel occupancy probability, the more likely it is a part of the shape. Let $g(\cdot)$ represents the 3D decoding network, then the final predicted 3D shape is

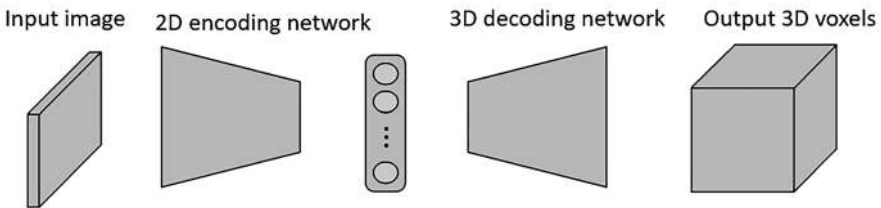


Fig. 1.5 Single-view reconstruction network structure

$$V^{\text{pred}} = g(h(I)) \quad (1.14)$$

This 3D shape is represented in the form of 3D voxels. The voxel value represents the probability of the 3D shape occupying this voxel position. Generally, a certain threshold (such as 0.5 or 0.4) is selected, and all voxels higher than the threshold constitute the 3D structure of the object.

In training, the mean square distance between the predicted voxels and the true values in the database can be established, and this distance can be optimized through backpropagation and gradient descent. By optimizing this distance function, the error between the predicted 3D voxel and the true value in the database in terms of the 3D structure can be reduced.

Using convolutional neural networks, input images can be encoded into semantic features and decoded into 3D voxels through a 2D encoding and 3D decoding network. However, often a single viewpoint cannot accurately express the shape information of an object. For example, a cup with a handle and a cup without a handle have no difference in their photos on the side without the handle, but they are different in 3D shape. It is clear that predicting the shape of an object from a single viewpoint is not enough, and the implementation of multi-view reconstruction in deep learning methods is crucial. Note: The multi-view here is different from the multi-view in traditional reconstruction methods. Traditional methods require very dense viewpoints to ensure that there is enough overlap area between adjacent frames for matching, while the multi-view here is still a very small number of viewpoints (usually less than 20 views). Deep learning methods only need to ensure that different viewpoints can observe the necessary structure of the object and do not need to ensure that the viewpoints can match each other. The understanding of the object's spatial position and geometric structure is implicitly represented in the deep structure of the neural network.

As mentioned earlier, for processing image sequence inputs, recurrent neural networks are a suitable network model. In multi-view reconstruction, it is customary to use 2D convolutional neural networks for feature encoding, and then send the encoded features into the recurrent layer. The recurrent layer combines the current encoding feature with the previous time's hidden layer state to update and output features. The final feature is decoded into 3D voxels by a 3D deconvolutional neural network.

A simple multi-view reconstruction network structure is shown in Fig. 1.6. The network inputs n viewpoint images $I_{1:n}$, these n images are passed through a 2D convolutional neural network with shared weights (denoted as $h(\cdot)$), and n encoded features $h(I_{1:n})$ can be obtained. Then these n features are input into the recurrent layer in sequence order (denoted as $f(\cdot)$). After the n th feature is input into the recurrent layer, the hidden layer state obtained is

$$S_n = f(h(I_n), S_{n-1}) \quad (1.15)$$

The final hidden layer state S_n is input into the 3D deconvolutional neural network to obtain the decoded 3D voxel. In the training process, the process is the same as the

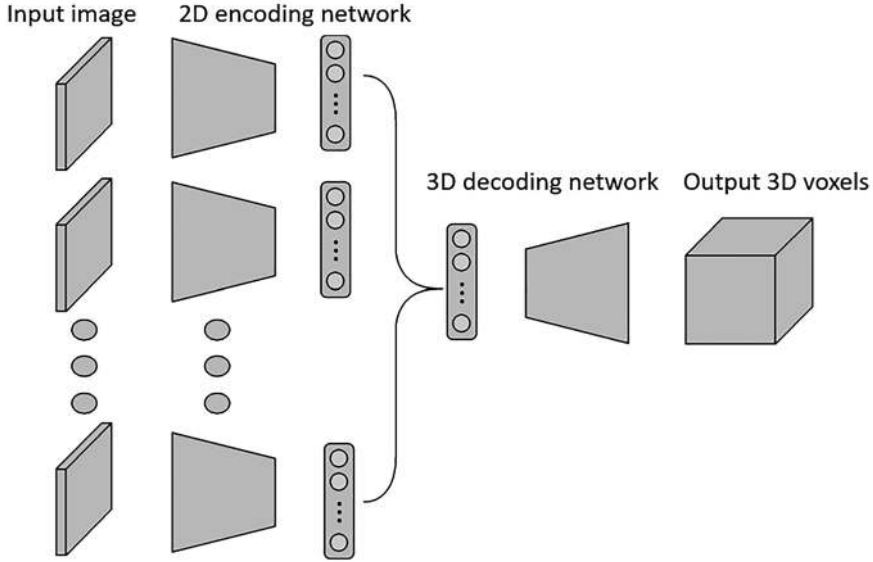


Fig. 1.6 Multi-view reconstruction network structure

convolutional neural network method of the single-view reconstruction network. As long as the image sequence is input and the 3D voxel predicted by the network is obtained, a loss function can be established for training. Using backpropagation and gradient descent to minimize this distance, the network training can be completed.

1.2.2.3 Limitations

The application of deep learning technology has opened up new ideas for 3D reconstruction. Compared with traditional methods, it uses fewer visual inputs and can use powerful predictive capabilities to handle complex environmental problems. However, there are still the following limitations:

First, deep learning technologies for 3D graphics mostly need to use 3D convolutional layers. Compared with 2D-pixel convolution, voxel cubic-level memory occupancy requires a lower resolution due to hardware limitations, resulting in the insufficient ability to express model details. Although octrees or point clouds, which occupy less space, can be used, the processing methods of these two types of data are more complex than those of spatially adjacent voxels, posing challenges to the design of deeper network models.

Second, as the number of viewpoints increases, the reconstruction effect of deep learning technology gradually enters a “bottleneck.” When the number of viewpoints is sufficient, traditional reconstruction methods are superior to deep learning technologies.

Third, the powerful learning ability of deep learning technology gives computers the ability to predict missing structures and occluded parts, but it also ignores the

meaning of visual information under each viewpoint in real scenarios. This leads to its generally excellent performance only on datasets similar to the training data, and its generalization is poor for more real scenarios.

Although deep learning technology is not as mature as traditional methods, nor as applicable as traditional methods, due to its inherent research value for breaking through the bottleneck of traditional methods, and the rapid development of hardware technology and related algorithms, it is destined that deep learning methods will gradually penetrate into traditional methods, and are likely to integrate and promote each other in the future.

1.2.3 Summary

This section explains the theoretical knowledge involved in 3D reconstruction, introduces representative algorithms from the two aspects of traditional image 3D reconstruction and reconstruction methods based on deep learning, and analyzes their limitations. In summary, traditional methods are affected by environmental conditions and the number of shooting angles. When the environment is ideal and there is enough visual information, its effect is better than that of deep learning methods. Deep learning methods, due to their powerful predictive ability, can be applied to 3D reconstruction when information is incomplete and the number of viewpoints is limited. However, due to the lack of use of geometric information and the excessive use of 3D data memory, the reconstruction results are not ideal and the generalization is poor.

From the comparison of traditional methods and deep learning methods, it can be seen that each has its own strengths and faces multiple challenges. In future research, both methods will receive attention from the academic and industrial communities and may be combined to complete a 3D reconstruction system that meets the requirements. This chapter will further guide the reader to understand the 3D reconstruction algorithm and the challenges and solutions it will face in practical applications based on specific technical cases in the following sections.

1.3 Technical Practice

1.3.1 Case Study: Depth Prediction Algorithm Based on Multi-Task Model

1.3.1.1 Task Description

The prediction of image depth values is a hot research topic in the field of computer vision and robotics. The construction of a depth map is an important prerequisite for 3D reconstruction. Obtaining the depth information of a scene is equivalent to obtaining the spatial distance of the pixel points. After the projection transformation

of the camera model, the 3D spatial position of the pixels can be calculated. Traditional methods mainly rely on manually annotating the depth of fixed points or predicting the depth based on the change in the camera's position for binocular positioning, but these methods are time-consuming and laborious and are limited by factors such as camera position and distribution probability, making it difficult to guarantee accuracy, resulting in the predicted depth map being difficult to complete subsequent 3D reconstruction work. To solve this problem, this technical case introduces a deep learning method based on a multi-task module, which builds a single-image depth prediction network based on a multi-task model for scene images. This network can simultaneously train and learn depth prediction, semantic segmentation, and surface normal estimation for three tasks. The network model includes a common feature extraction module and a multi-task feature fusion module, which can ensure the independence of each feature while extracting common features, and improve the structure of each task while ensuring the accuracy of depth prediction.



Chapter Knowledge Points

Depth Map: In 3D computer graphics, a depth map is an image or image channel that contains information related to the distance of the surface of scene objects from the viewpoint. In this case, a depth map is similar to a grayscale image, except that each pixel value is the actual distance from the sensor to the object. Typically, RGB images and depth maps are registered, so there is a one-to-one correspondence between pixels.

Semantic Segmentation: In computer vision, the task of semantic segmentation is to segment images or point clouds, distinguishing different segmented objects. When using semantic segmentation, it divides the image or point cloud into semantically meaningful parts and then labels each part as one of the predefined classes semantically.

Normal Vector: The normal vector is a concept in spatial analytic geometry. The vector represented by a line perpendicular to the plane is the normal vector of that plane. The normal of a 3D plane is a 3D vector perpendicular to that plane. The normal at a point P on a surface is the vector perpendicular to the tangent plane at that point. The normal vector is applicable to analytic geometry. Since there are countless lines in space perpendicular to a known plane, there are countless normal vectors (including two unit normal vectors) for a plane.

Binocular matching: Matching with two binocular cameras. For a feature point on an object, images of the object are taken with two cameras fixed at different positions, and the coordinates of the point on the image planes of the two cameras are obtained, respectively. As long as the precise relative positions of the two cameras are known, the coordinates of the feature point in the coordinate system of one fixed camera can be obtained by geometric methods, thus determining the position of the feature point.

1.3.1.2 Related Work

1. Monocular Image Depth Prediction

Before the emergence of machine learning, two main methods were used for monocular image depth prediction: One is the traditional method, which uses manual annotation to locate the depth of similar images, and then makes reasonable geometric assumptions about the scene, analyzing the depth values of each part based on a probabilistic model. The other method is used by Saxena et al. [34, 35] using Markov random fields (MRF), regularizing the depth map, and then predicting the corresponding depth values of pixels on the plane through plane coefficients and random fields. Because the formula includes parts that constrain adjacent pixels, this method can infer depth information by analyzing local and global features in the image.

With the widespread use of deep learning methods in the field of computer graphics, Laina et al. [36] used fully convolutional networks (FCN) for monocular depth estimation and promoted the improvement of effects by deepening the network structure. FCN does not limit the resolution size, so no image post-processing is needed. Cao et al. [37] treated the depth prediction problem as a fine semantic segmentation. First, the depth values in the image were discretized, and then a residual network was trained to estimate the classification corresponding to each pixel. Finally, the conditional random field (CRF) was used for processing optimization based on its corresponding probability distribution. Garg et al. [38] were the first to achieve unsupervised monocular depth prediction, using stereo images without the need for depth value data. By using the relationship between the left eye image, the right eye image, and the depth image, they first input the left eye RGB image to predict a blurry depth image, then restore the right eye image through the depth image and the original image, and calculate the loss between the right eye image and the true value to optimize the network. Zoran et al. [39] were more interested in the relative depth relationship in the image, using the point-matching relationship in the binocular system to estimate the depth value, in order to achieve unsupervised or weakly supervised purposes. This algorithm first uses the correspondence between points in stereo images to interpolate two predicted depth images for optimization, in order to obtain more accurate depth. Casser et al. [40] proposed a more effective method, that is, by modeling moving objects, higher quality depth estimation results are produced. In addition, a seamless online optimization method is used to make the training effect better, improve the robustness of prediction, and can be transferred to different datasets.

Although convolutional neural networks can achieve good results in depth prediction tasks, there are still some problems. First, the network structure used for depth prediction is mostly used to solve semantic segmentation problems. Although it can achieve a high accuracy rate, it is difficult to express the structural nature of depth images; second, the small number of depth image datasets and the noise problems they have can also affect the robustness of prediction.

2. Depth Prediction and multitasking

Looking at the work of depth prediction, it is inseparable from the methods of semantic segmentation. Similar feature extraction methods make depth prediction can be trained together with other tasks, such as semantic segmentation, which can extract the same features of each task and use the mutual relationship of each task to promote each other. Recently, several joint task learning methods for such tasks have shown satisfactory results, improving prediction accuracy by forming multi-task driving through combining task-related information. Eigen et al. [41] proposed that depth prediction can be used with tasks such as semantic segmentation and surface normal prediction to train a network together. The model proposed by Mousavian et al. [42] trained the depth value prediction and semantic segmentation tasks together, first training the two tasks together in the first stage, and then using the loss function to converge training separately in the second stage to adjust the entire network to form optimization. Then, CNN and CRF are combined to optimize the results using semantic labels and depth value information. Wang et al. [43] also combined CRF and CNN to build a model, proposing a framework that can combine depth with various other visual information, and by deriving well-designed energy functions, the entire framework can be trained through backpropagation. Godard et al. [44] trained four basic visual problems of single-view depth prediction, camera pose prediction, optical flow information, static scene, and moving area semantic segmentation together through geometric constraints. Zhang et al. [45] processed RGB images into feature maps of different scales, obtained different depth information and semantic information through the decoder, and improved accuracy through alternating feature fusion. Zhang et al. [46] proposed a “pattern-affinitive propagation, PAP” framework to simultaneously predict depth, surface normals, and semantic segmentation tasks. Through two types of propagation, the network can better adapt to the differences between different tasks, and the learning of each task can be standardized and enhanced by supplementing task-level capabilities.

The main problem with current multi-task models is that the feature fusion method only uses parameter sharing, and does not show the help of common features to tasks and the characteristics between tasks.

1.3.1.3 Technical Methods

By jointly predicting tasks related to depth prediction, the interrelated information between the two can be utilized to mutually promote and improve results. The fundamental reason is that for the same semantic information, the network will provide the same parameters, and sharing these parameters will make some common information more prominent, avoiding the omission of some feature extraction due to data reasons in a certain task. Among them, semantic segmentation and surface normal prediction are two tasks closely related to depth prediction. On the one hand, semantic segmentation can effectively extract features to distinguish different objects, and through the characteristics of gradient transformation between objects,

it can be associated with objects in depth prediction. On the other hand, semantic segmentation can classify different objects, making the prediction of depth values more dependent on the unique gradient changes of the object, and better distinguish the connection between objects similar to the background. The impact of surface normals on depth prediction is more direct, and there is a geometric relationship between the two that can be converted to each other, and its mathematical relationship is

$$n_{ix}(x - x_i) + n_{iy}(y - y_i) + n_{iz}(z - z_i) = 0 \quad (1.16)$$

For each pixel i , giving its normal vector (n_{ix}, n_{iy}, n_{iz}) and the corresponding depth z_i , a tangent plane can be determined, and adjacent pixels are likely to be in the same tangent plane. The surface normal is another representation of the depth gradient, which can complement each other, and joint training can better extract features to complete the recovery of depth prediction.

To achieve this purpose, this technical case will carry out network training through two modules, one module is the common feature extraction module, responsible for extracting the features of each task, and can recover rough depth images, semantic segmentation images, surface normal images. The other module is the multi-task feature fusion module, which is responsible for fusing the features extracted by the first module for multi-tasking. The network model will distinguish the common semantic features of each task and the unique semantic features of each task, making the final recovered image more structured.

Figure 1.7 shows the process of the entire algorithm and the information of the two modules. The overall network input is a single RGB image, and the output is the depth image, semantic segmentation image, and surface normal image corresponding to this image. The network integrates the common feature extraction module and the multi-task feature fusion module into a unified structure and completes the end-to-end generation in unified training.

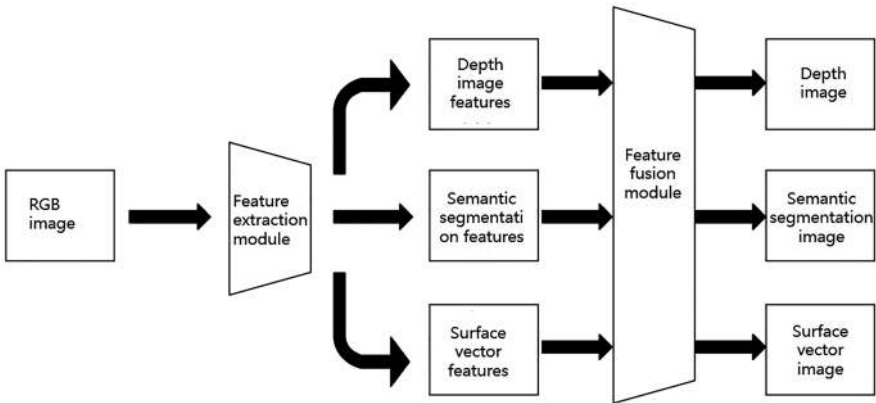


Fig. 1.7 Process of multi-task-based deep prediction algorithm



Sample Code

Input: RGB image

Output: Depth prediction, semantic segmentation, surface normal prediction

1. Use the multi-task feature extraction module to extract features d, s, n from the RGB image separately.
2. The multi-task feature fusion module inputs d, s, n , and obtains depth prediction, semantic segmentation, and surface normal, respectively.

Figure 1.7 and the sample code outline the entire algorithm process. The following will elaborate on the network structure and training process in terms of the feature extraction module (including single encoders and multiple decoders) and the cross-task feature fusion module (including multiple encoders and multiple decoders), and finally evaluate the model through experimental comparison.

1. Common Feature Extraction Module

The purpose of the common feature extraction module is to implement common feature extraction for depth prediction, semantic segmentation, and surface normal prediction, as well as task differentiation in the first stage. This module first encodes the input picture through the encoder to form a one-dimensional feature, then decodes it into the size and channel required for the three tasks through three decoders, then combines the features of each scale through the feature connection method, and finally can build a loss function based on the image output by this module and the true value image in the dataset, updating the network parameters to ensure that the encoder extracts the common features of the three tasks.

This module consists of four parts: Encoder, multi-dimensional decoder, multi-scale feature fusion module, and refinement module (as shown in the network model in Fig. 1.8). The encoder, composed of four convolutional layers, is responsible for extracting features of multiple scales such as $1/4$, $1/8$, $1/16$, and $1/32$. The decoder uses four upsampling modules to gradually enlarge the final features of the encoder while reducing the number of channels. The multi-scale feature fusion module uses upsampling and channel concatenation to integrate features from the encoder at four different scales. Corresponding to the encoder, the outputs of the four layers of the encoder (each with 16 channels) are upsampled by $\times 2$, $\times 4$, $\times 8$, and $\times 16$, respectively, to have the same size as the final output. This upsampling is done by channel concatenation, and then further transformed through a convolutional layer to obtain an output with 64 channels. The main purpose of the multi-scale feature fusion module is to merge different information from multiple scales into one, thus preserving the lower layer output of the encoder with more refined spatial resolution, which helps to recover the

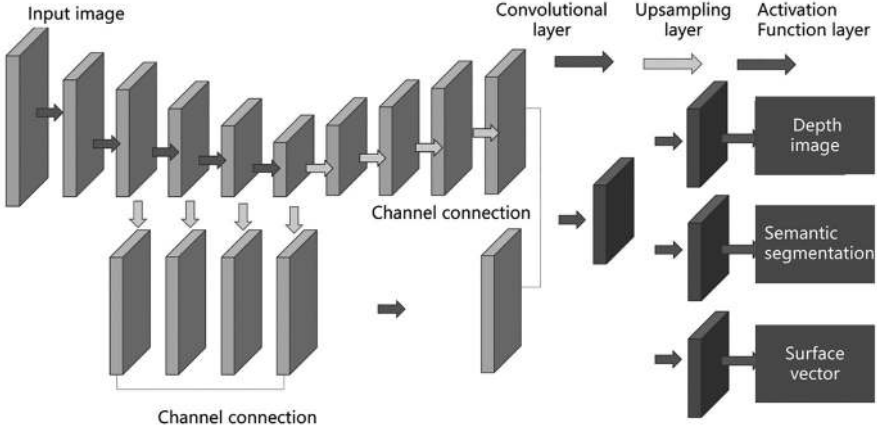


Fig. 1.8 This figure shows the structure of the feature extraction module (see the color picture before the text)

detail information lost due to multiple downsampling. Finally, the refinement model is used to adjust the output size and channel number of the image, corresponding to three tasks using three convolutional layers, so that the output channel number is restored to one channel for depth images, one channel for semantic segmentation images, and three channels for surface normals, which is convenient for loss calculation and backpropagation.

Training process: This module belongs to a single-input multi-output network. During the training process, the network needs to establish a loss function based on the RGB image, depth image, semantic segmentation map, surface normal map in the dataset, and the corresponding images predicted by the network, to update the network parameters, iterate until the loss function converges, and then obtain the well-trained network.

Since it needs to handle three tasks at the same time and ensure the relevance between each task, the loss function is divided into three parts. The feature extraction loss function is

$$L_{\text{task}} = L_{\text{depth}} + L_{\text{seg}} + L_{\text{normal}} \quad (1.17)$$

The roles and purposes of each part of the loss function are as follows:

- (a) **Depth Image Loss Function:** As shown in eq. (1.18), this function is the sum of the L1 loss function and the gradient loss function. For each pixel point i , the corresponding predicted depth and real depth are d_i and D_i , respectively. The L1 loss function can constrain the difference between d_i and D_i , which is the main part of this loss function and provides a guarantee of accuracy.

$$L_{\text{depth}} = L_1 + L_{\text{grad}}$$

$$L_1 = \frac{1}{n} \sum_{i=1}^n |D_i - d_i| \quad (1.18)$$

- (b) **Semantic Segmentation Loss Function:** Formally, it is a cross-entropy function, which is a commonly used loss function in semantic segmentation problems. It is often used to describe the distance between two probabilities. In this technical case, it constrains the probability of the predicted semantic category of objects in the image. Its expression is

$$L_{\text{seg}}(S, \hat{S}) = - \sum_{j=1}^{\text{classes}} s_j \ln \hat{s}_j \quad (1.19)$$

For each pixel s and $\hat{s}$ in the predicted semantic segmentation image S and the true image $\hat{S}$, there are nonzero or one element in the class categories, and only one s_j value is 1, so the cross-entropy loss only cares about the prediction probability of the correct category. As long as its value is large enough, it can ensure the correctness of the classification result.

- (c) **Surface Normal Loss Function:** That is, measuring the accuracy of the estimated depth map surface normal (n_i^d) relative to its GT surface normal (n_i^g). The surface normal loss is also calculated based on the depth gradient, but it measures the angle between the two surface normals, so this loss is very sensitive to the depth structure and can enhance the consistency of the predicted structure. The expression is

$$L_{\text{normal}} = \frac{1}{n} \sum_{i=1}^n \left(1 - \frac{n_i^d, n_i^g}{\sqrt{n_i^d, n_i^d} \sqrt{n_i^g, n_i^g}} \right) \quad (1.20)$$

- (d) **Gradient Loss Function:** It can constrain the gradient changes on the x -axis and y -axis ($g_x(e_i)$ and $g_y(e_i)$ are the gradient losses on the x -axis and y -axis, respectively). Since the depth at the boundary of the object is usually discontinuous, the gradient change is calculated to sensitively detect edge information. It should be noted that the gradient loss is a different type of error from the previous depth loss, so it needs to be weighted to train the network. Its expression is

$$L_{\text{grad}} = \frac{1}{n} \sum_{i=1}^n |g_x(e_i)| + |g_y(e_i)| \quad (1.21)$$

2. Multi-task Feature Fusion Module

This module is responsible for processing the data from the common feature extraction module. Through this module, the common features are more finely fused and differentiated, thereby extracting the specific common features and

differentiated features of each task, improving the accuracy of each task, and enhancing the structural consistency of each task. Since the common feature extraction module only extracts common features through parameter sharing, it cannot accurately represent the commonality of the three tasks in this paper, so the design of this module can ensure the universality of multi-task feature extraction.

The multi-task feature fusion module consists of two parts: The first part is the multi-input feature fusion module, which is responsible for fusing the multi-task features output by the previous module. The network used is a densely connected U-net. The second part is the feature decoding part, which is similar to the decoder part of the previous part, and is a multi-output decoder, so it is not detailed. The overall network structure of this module is shown in Fig. 1.9.

This technical case adds a dense connection method based on the original U-net network, which can effectively enhance the feature extraction ability of the multi-input mode. To implement this dense connection mode, first create an encoding path composed of multiple streams, each stream processes the image form of different tasks from the previous module. The main purpose of using separate streams for different image forms is to disperse information that would have been fused early, thereby limiting the network's ability to learn complex relationships between patterns. From the network structure, it can be seen that the three tasks are first encoded separately through the U-net encoder, but the difference is that interaction will occur when passing through different convolution layers. For example, after the image of task 1 undergoes a convolution pooling operation, the obtained pooling feature will be combined with the secondary pooling feature of task 2, and after the convolution layer, it will be combined with the pooling feature of task 3. This can make features flow between tasks and ensure the commonality of features. In the decoder part, the obtained common features are first upsampled to obtain a common upsampled feature, and then this upsampled feature is combined with the previous pooling feature for decod-

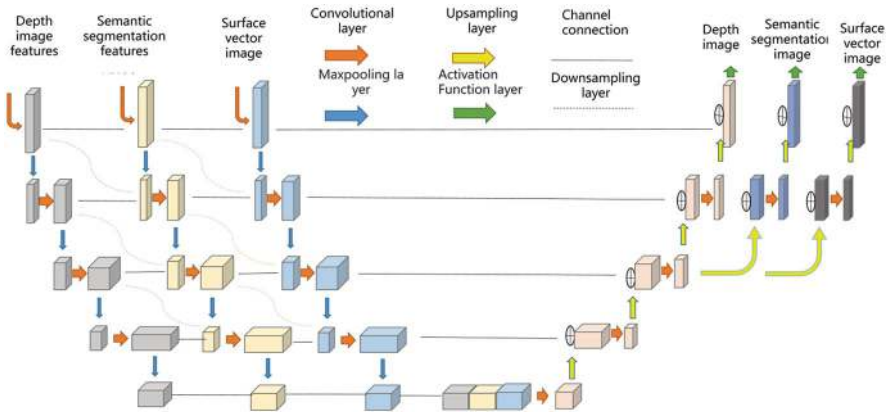


Fig. 1.9 Multi-task feature fusion module structure (see color figure before the text)

ing, and it is sent to the decoder with the pooling features of three tasks of different scales, and connected with the previously extracted features of each task through the upsampling layer, restoring the original shape of each task, and finally comparing the restored depth image, semantic segmentation image and surface normal image with the true value in the dataset, calculating the gradient to update the parameters in the network.

When training the network, this module can be trained together with the common feature extraction module to form an end-to-end neural network, that is, the input is a single RGB image, and the output is the corresponding depth image, semantic segmentation image and surface normal image. Since the three tasks are the same as the common feature extraction module, the same loss function is used for constraint, so it is not detailed.

1.3.1.4 Results Display

The dataset used in this technical case is built upon the NYU Depth V2 dataset [47]. It uses 1449 sets of RGB images, semantic segmentation images, and depth images for training and testing. The surface normal images are converted from depth images using the method proposed by Chen [8], and the resolution of the images is uniformly converted to 320×240 , consistent with the network input. Among them, 795 sets are used for training, and 654 sets are used for testing. Each set of training images obtains four sets of training sets to enhance the robustness of training through a 10-pixel shift up, down, left, and right.

Figures 1.10, 1.11, and 1.12, respectively, show the partial visualization results of the three tasks of depth prediction, semantic segmentation, and surface normal prediction in this technical case. Figure 1.10 includes the FRCN [36] method. From the visualization results, it can be seen that the results of this technical case can basically output a good result for each task.

1.3.1.5 Summary

In this technical case, a monocular depth prediction network based on a multi-task model is proposed. By training with semantic segmentation and surface normal prediction tasks, the mutual connection between depth prediction, semantic segmentation, and surface normal prediction tasks is established, thereby improving the training effect of the network. This algorithm enhances the structural nature of depth prediction through a common feature extraction module, and further distinguishes the unique features and connections between tasks through a multi-task feature fusion module, so that the prediction effect of each task can be further improved. Experiments show that this technical case has improved the effect of each module, and the multi-task feature extraction and cross-task feature fusion have promoted the prediction of network depth values. Comparative experiments also show that this algorithm has superiority.

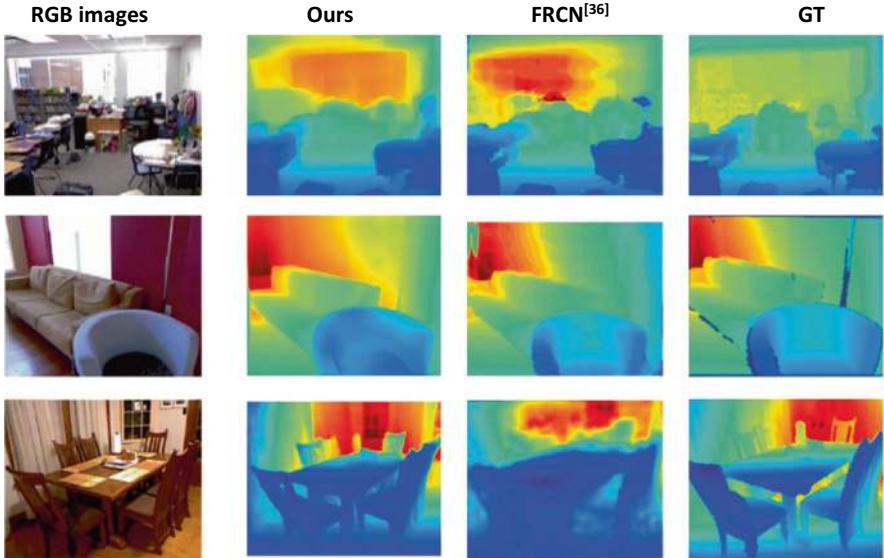


Fig. 1.10 Visualization results of depth prediction

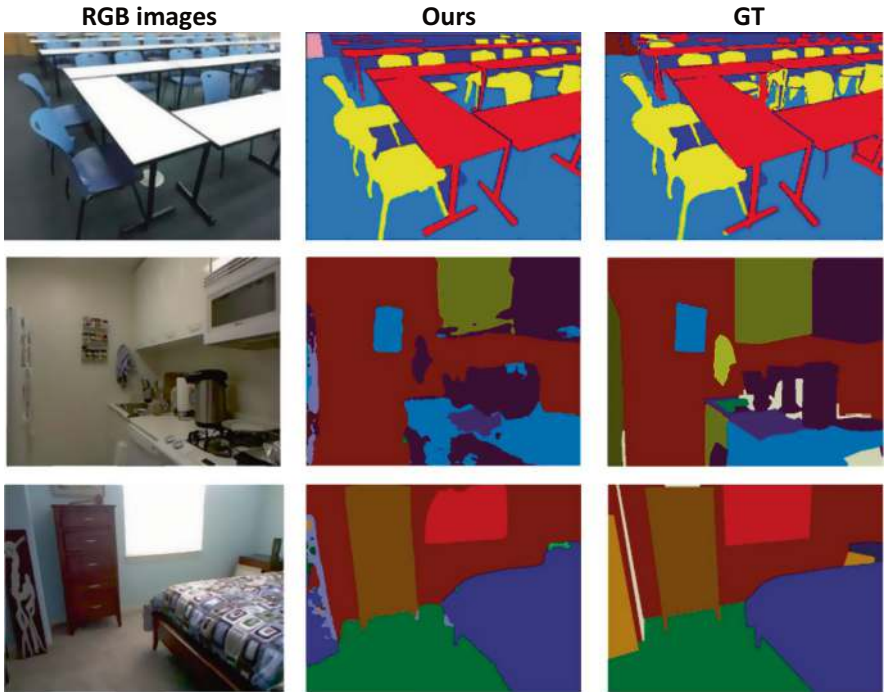


Fig. 1.11 Visualization results of semantic segmentation (see color picture before the text)

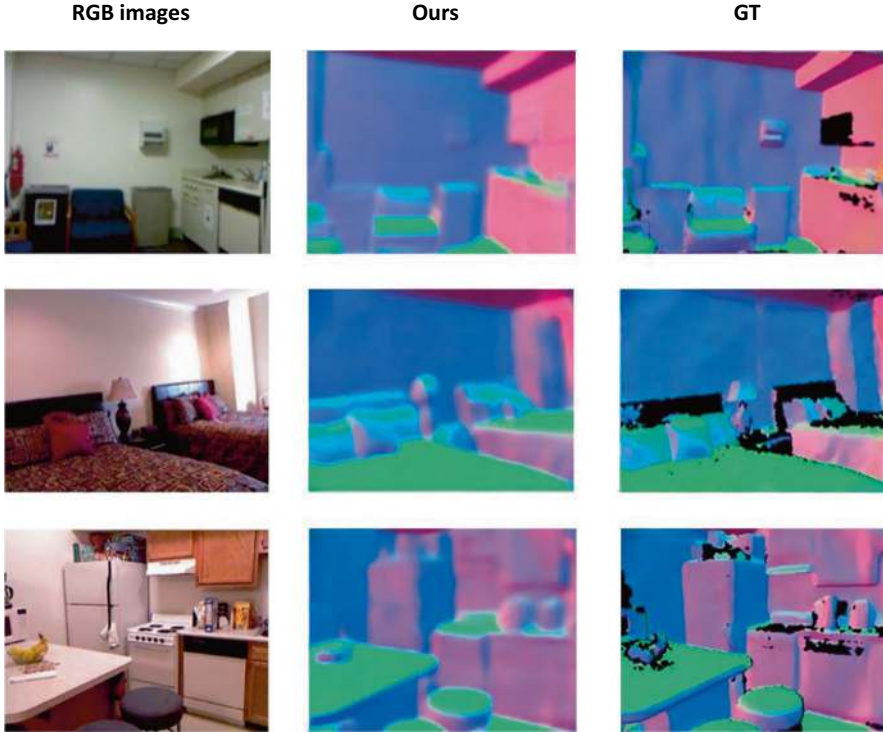


Fig. 1.12 Visualization results of surface normal prediction (see color picture before the text)

1.3.2 Case Study: Active 3D Object Reconstruction and Completion Optimization

1.3.2.1 Task Description

The goal of this case is to implement an active 3D object reconstruction and completion optimization method for objects. The active 3D object reconstruction algorithm refers to reference [29], and the completion optimization method uses a combination of the Variational Autoencoder network VAE [48] and the Generative Adversarial Network GAN [49]. The entire process specifically includes the following three modules: viewpoint dynamic prediction module, reconstruction module, and reconstruction completion optimization module; These three modules are used to autonomously plan the scanning view of the object, perform 3D reconstruction of the object, and complete and optimize the reconstruction results. This technical case is designed to solve the technical problems of traditional 3D object reconstruction being easily affected by environmental factors, low efficiency, and difficulty in achieving autonomy. It designs a 3D object reconstruction framework and software platform based on deep learning technology, which can dynamically plan the

scanning view for a given object, and combine pictures from different views to complete the construction of the 3D object model. Finally, the recently popular generative adversarial network is used to complete and optimize the reconstructed 3D model.



Chapter Knowledge Points

Reinforcement Learning (RL): Used to describe and solve the problem of an intelligent agent interacting with the environment and learning policies to maximize rewards or achieve specific goals.

Variational Autoencoder (VAE): Generates the mean and variance of the target distribution through the encoding process, then restores the target sample distribution through sampling techniques, and finally adjusts the parameters based on the distance between the restored distribution and the real distribution.

Generative Adversarial Nets (GAN): Generates models through a mini-max game between the generative network and the discriminative network. The name GAN contains the concept of adversarial, to reflect this concept, in addition to the generative model, there is another discriminative model to help the generative model better learn the conditional distribution of observed data.

3D Model Completion: In the process of object modeling, there are inevitably situations such as occlusion and jitter, which cause the loss of 3D data model information and incomplete model structure. It is necessary to predict and fill in the missing parts of the model. 3D model completion is to calculate and predict the geometry of the missing parts through the known 3D structure to obtain a relatively complete 3D reconstruction.

1.3.2.2 Related Work

With the development of SLAM technology, the method of 3D reconstruction of indoor scenes is becoming more mature. 3D reconstruction generally includes three parts: first, use a handheld camera to scan the object from multiple views, then extract features from the scanned multiple frames, match them, estimate the camera pose, and finally, complete the mapping of 2D pixels to 3D coordinate points through stereo vision technology to get the final reconstructed model. However, in previous work, the scanning of the object often covers every local structure of the object, which is inefficient and also makes the scanning path planning for a certain object not applicable to other types, bringing challenges to the autonomy of scanning. Therefore, an efficient reconstruction method that can autonomously plan scanning views during the reconstruction process is a technical difficulty to overcome, and it is also the motivation for this technical case. The following briefly introduces the relevant background technology in this field.

1. 3D Reconstruction Technology

The University of Washington and Microsoft Labs [10] developed a real-time visual SLAM system based on SIFT (Scale Invariant Feature Transform) feature matching positioning. Through this real-time system, a 3D map of the scene can be established. Subsequently, many works have made improvements in real-time performance and pose estimation, including RGB-D SLAM algorithm [50], KinectFusion [6, 7], and BundleFusion [13], basically meeting the real-time requirements of scene reconstruction and user interaction.

However, these algorithms inevitably face the following problems: first, the algorithm requires dense views, but in actual calculations, most views are skipped, making the efficiency of information acquisition low and unable to be applied in scenes with a lot of occlusion; second, the algorithm needs to assume that there is no mirror reflection in the scene, and it has rich textures to meet the feature extraction of the picture; third, despite the use of various optimization strategies, the problem of camera registration error accumulation still exists.

To address the above issues, Choy et al. [27] introduced deep learning into the work of 3D reconstruction of objects, proposing a 3D recurrent neural network. By constructing a 3D hidden layer state, it receives pictures taken from multiple views, while implicitly representing the current geometric structure being reconstructed. This establishes a 3D reconstruction framework that combines single-view and multi-view, showing superior performance to traditional methods in both single-view and multi-view (less than 20 views).

The application of deep learning technology has opened up new ideas for 3D reconstruction. Compared with traditional methods, it uses fewer visual inputs and can handle complex environmental factors, and the combination of traditional methods and deep learning ideas may be a new direction in the field of 3D reconstruction.

2. View Planning Methods

A robot, facing an unknown object from an arbitrary view, actively predicts the next observation view based on the current observation and estimates the view based on the observation of the next view. Such continuous view can help robots actively perceive interesting information and complete corresponding visual tasks. Actively perceiving the surrounding environment through consumer-grade cameras and digitizing the acquired environmental information is a technical challenge to be overcome in the field of robotics. In previous work, some rule constraints were often used to maximize the information received. Common methods include uncertainty reduction [51], Monte Carlo sampling [52], and Gaussian process regression [53].

In recent years, deep learning methods have also been used to predict the selection of views, such as using deep belief networks [24] to estimate the information gain of views, and vision attention models based on reinforcement learning [54], etc. The work most relevant to this technical case is the recognition of 3D objects by Xu et al. [55]. This work combines multi-view convolutional neural networks [56] and recurrent attention models [54] to actively acquire depth data during the object recognition process and introduces a spatial transfer network in subsequent work to achieve end-to-end learning. The active perception

of this scheme is reflected in the ability to predict the next best view based on the visual observation obtained from the current view, achieving view prediction based on visual feedback, enabling robots to autonomously complete object recognition.

3. 3D Model Completion Optimization

For a long time, experts have tried to use geometric principles to complete 3D models whose structures have been damaged, including Laplacian smoothing [57], Poisson surface reconstruction [58], and prediction of locally occluded parts using symmetry [59]. Although the above methods can achieve good results, the predefined rules used by these methods fundamentally limit the shape space, and the models used for learning and detection basically need to be manually made or processed through multiple scans in a real environment. By combining data prior knowledge and symmetry detection, many works [60–62] have found corresponding 3D models in the database based on given partial information missing incomplete data and completed them through nonrigid transformation and alignment. However, these methods rely too much on the database, requiring the database to be large enough and the data types to be diverse, leading to poor generalization, and the details of the model to be optimized are easily lost.

Generative network models have also seen rapid development. For example, the recently widely noticed Generative Adversarial Networks (GAN) [49] incorporate adversarial discriminators into the generative model process, enabling the input random vectors to obtain better-generated objects. A common method [48] is to transform the input into latent vectors as the input of the GAN and make the generative model more realistic through the adversarial process of the generative network and the adversarial network. This technical case uses a network model built with deep learning to learn the latent information and internal connections of the 3D model and integrates the generative adversarial idea into the completion optimization process of the 3D model, using their associative ability to learn the object representation method of 3D voxels. This method does not require manual feature extraction of the model, nor does it require database retrieval, but obtains the global structure and local details of the 3D model through the generative ability of the network.

1.3.2.3 Technical Method

This technical case includes a viewpoint dynamic prediction module, a object reconstruction module, and a reconstruction completion optimization module. Details are as follows:

1. Viewpoint Dynamic Prediction Module

- (a) **Module Input:** Use an RGB-D camera to collect indoor object information. At any random viewpoint v_0 around the object, take a color photo with the RGB-D camera, use the three RGB channels, and compress the resolution to 64×64 , to get a $64 \times 64 \times 3$ tensor I_0 . The random viewpoint v_0 and the picture tensor I_0 together constitute the input of the module.

- (b) **Module Architecture:** The viewpoint dynamic prediction module is a two-branch neural network with different states, inputs, and outputs at different time steps. At a certain time t , the first branch network is a fully connected layer f_{view} , which takes the random viewpoint v_t as input and calculates the corresponding viewpoint feature $f_{\text{view}}(v_0)$; the second branch network is a multi-layer convolutional recurrent neural network f_{enc} , which becomes the encoding network, responsible for encoding the input picture tensor I_t into low-dimensional features $F_t = f_{\text{enc}}(I_t, S_{t-1}^{\text{enc}})$, where S_{t-1}^{enc} is the storage state of the recurrent layer at the previous time step, and its value is the unit matrix when $t = 1$.

Then, the features extracted by the two branch networks are multiplied element-wise, and after the processing of another recurrent layer f_{gru} and a fully connected layer f_{fc} , the final feature vector is obtained as in eq. (1.22):

$$F = f_{\text{fc}} \left(f_{\text{gru}} \left(f_{\text{view}}(v_0) \right) * f_{\text{enc}} \left(I_t, S_{t-1}^{\text{enc}} \right), h_{t-1}^{\text{gru}} \right) \quad (1.22)$$

where h_{t-1}^{gru} is the storage state of the recurrent layer at the previous time step, and its value is the unit matrix when $t = 1$.

Finally, the feature vector F is processed by the S function (sigmoid) to obtain the final predicted viewpoint, which is used as the input viewpoint v_{t+1} for the next time step. At the same time, the RGB-D camera is used to obtain the picture at this viewpoint, to get the input tensor I_{t+1} for the next time step.

- (c) **Training Method:** The training of the neural network in the viewpoint dynamic prediction module uses reinforcement learning methods. At each time step, the reward is calculated according to eq. (1.23):

$$r'_{\text{cons}} = \text{IoU}(\hat{V}_t, V) - \text{IoU}(\hat{V}_{t-1}, V) \quad (1.23)$$

where $\hat{V}_t$ is the 3D reconstruction model at time t ; V is the real model in the database; and IoU is the proportion of overlapping elements of the two models to all elements.

After accumulating the rewards of each time step, the network is trained using a policy estimation-based optimization method; specifically, the gradient of eq. (1.24) is calculated using the gradient descent method, and then the network parameters are iteratively updated in the direction of gradient descent to obtain the neural network that predicts the optimal viewpoint.

$$L_{rl} = \sum_{t=1}^T -\log \left(p(v_t^p | I_t) \right) * R \quad (1.24)$$

Here, R is the cumulative reward for all time steps, and $p(v_t^p | I_t)$ is the probability of predicting this viewpoint.

2. Reconstruction Module

- (a) **Module Input:** Using the RGB-D camera to take pictures under the viewpoint predicted in the viewpoint dynamic prediction module, each frame of

the picture is processed in the same way as in the viewpoint dynamic prediction module according to the time step order, and the ordered picture tensor sequence $\{I_0, I_1, \dots, I_n\}$ is obtained as the module input.

- (b) **Module Architecture:** The reconstruction module is a recurrent neural network, which includes an encoding network and a decoding network. The encoding network f_{enc} uses the encoding network (the second branch network) of the viewpoint dynamic prediction module, which is a multi-layer convolutional recurrent neural network, responsible for encoding the low-dimensional features F_t of the picture tensor I_t input at time step t . The decoding network f_{dec} consists of multiple layers of 3D deconvolution recurrent layers, which upsample the low-dimensional picture features F_t to obtain the 3D voxel $V_t = f_{\text{dec}}(F_t, S_{t-1}^{\text{enc}})$, where S_{t-1}^{enc} is the storage state of the recurrent layer at the previous time step, and its value is the unit matrix when $t = 1$.

For a picture sequence composed of t time steps, it is input into the viewpoint dynamic prediction module in the order of time steps, and the 3D voxel predicted at the last (i.e., the t -th) time step is the 3D reconstruction result of the object.

- (c) **Training Method:** The reconstruction module is trained using backpropagation and stochastic gradient descent. For a batch sample, the error between the network's predicted result and the database's true result is calculated according to eq. (1.25), and the gradient of the error is calculated. According to the backpropagation of the neural network, the network parameters are gradually updated in the direction of gradient descent until convergence.

$$L_{\text{vox}} = \|V_{\text{pre}} - V\|^2 \quad (1.25)$$

Here, V_{pre} and V represent the voxel model predicted by the reconstruction module network and the corresponding voxel model in the database, respectively.

3. Reconstruction Completion Optimization Module

- (a) **Module Input:** The reconstruction result V_{pre} output by the reconstruction module is used as the input, and the input is in the form of a 3D body, where each voxel value represents the probability of the 3D shape occupying that position in space.
- (b) **Module Architecture:** The reconstruction completion optimization module includes an encoding network g_{enc} , a generation network g_{dec} , and a discrimination network g_{dis} . The encoding network corresponds to the encoding part of the variational autoencoder [48], consists of 5 3D convolution layers with a stride of 2, and uses the activation function LeakyReLU. The output of the network is the mean vector m and variance vector var of the input model's latent feature distribution. The generation network corresponds to the decoding part of the variational autoencoder, consists of 5 3D deconvolution layers with a stride of 2, and uses the activation function ReLU. The final output is the probability V_{gen} of the 3D shape occupying the grid through the sigmoid activation function. The discrimination network is similar to the

encoding network in structure, but it connects a fully connected layer and a sigmoid activation function at the end of the network to judge the authenticity of the generated model.

- (c) **Training Method:** The reconstruction completion optimization module is trained using backpropagation and stochastic gradient descent. For a batch sample, which includes data pairs of incomplete data V_{inc} and complete data V_{com} , they are fed into the encoding network g_{enc} , generation network g_{dec} , and discrimination network g_{dis} . The generation loss function (eq. 1.26) is used to constrain the generation quality, the KL divergence (eq. 1.27) is used to constrain the difference between the input feature and the output distribution, and the discrimination loss function (eq. 1.28) is calculated. The final loss function L_{total} can be obtained by weighted summation. Through backpropagation and gradient calculation, the parameters are updated according to the learning rate of each network until convergence.

$$L_{\text{gen}} = \frac{1}{n} \sum_n \left(g_{\text{dis}} \left(g_{\text{dec}} \left(g_{\text{enc}} \left(V_{\text{inc}} \right) \right) \right) - 1 \right)^2 \quad (1.26)$$

$$L_{KL} = -\frac{1}{2n} \sum_n m^2 + \text{var} - \log(\text{var}) - 1 \quad (1.27)$$

$$L_{\text{dis}} = -\frac{1}{n} \sum_n \left(g_{\text{dis}} \left(V_{\text{com}} \right) - 1 \right)^2 + g_{\text{dis}} \left(g_{\text{dec}} \left(g_{\text{enc}} \left(V_{\text{inc}} \right) \right) \right)^2 \quad (1.28)$$

$$L_{\text{total}} = \lambda_{\text{gen}} L_{\text{gen}} + \lambda_{KL} L_{KL} + \lambda_{\text{dis}} L_{\text{dis}} \quad (1.29)$$

where λ_{gen} , λ_{KL} , and λ_{dis} represent the weights corresponding to the three loss functions, which are hyperparameters that need to be manually adjusted.

The following will further explain this technical case in combination with specific implementation methods, including three parts: training the network model, reconstruction process, and reconstruction completion optimization process.

- **Training the Network Model:** First, construct two large databases, one of which, X, should contain the object's 3D voxel model and multi-view pictures for training the network model of the viewpoint dynamic prediction module and the object reconstruction module. The other database, Y, should contain voxel models of incomplete shapes and corresponding complete shape voxel models. Next, build the network model according to Figs. 1.13 and 1.14 (referred to as Network A and Network B, respectively). Then, multi-thread the training data in database X into batches to the network A to be trained, and calculate the errors in the viewpoint dynamic prediction module and the object reconstruction module according to formulas (1.24) and (1.25). Update the network parameters iteratively using the gradient descent optimizer according to the method of backpropagation until the number of iterations meets the requirements, and complete the training of the viewpoint dynamic prediction module and the object reconstruction module network. Finally, multi-thread the

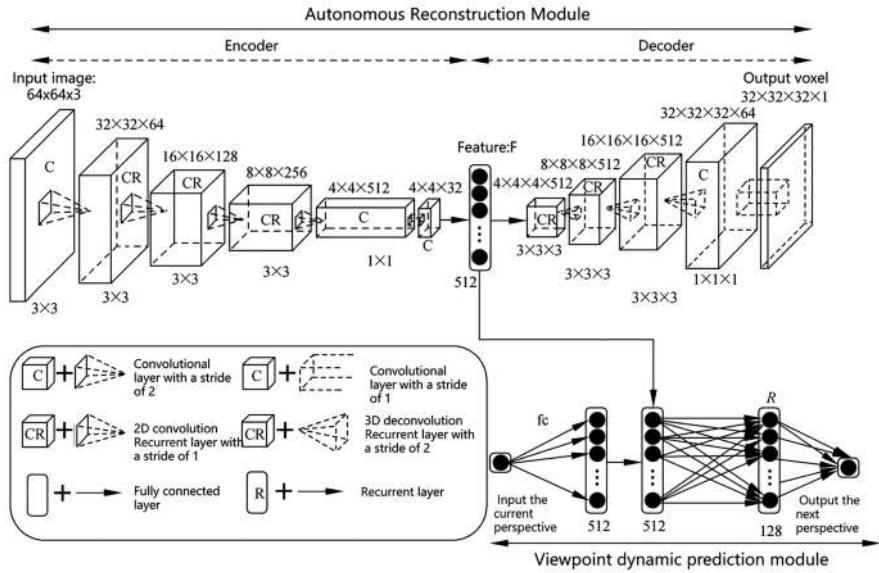


Fig. 1.13 Reconstruction module network architecture diagram

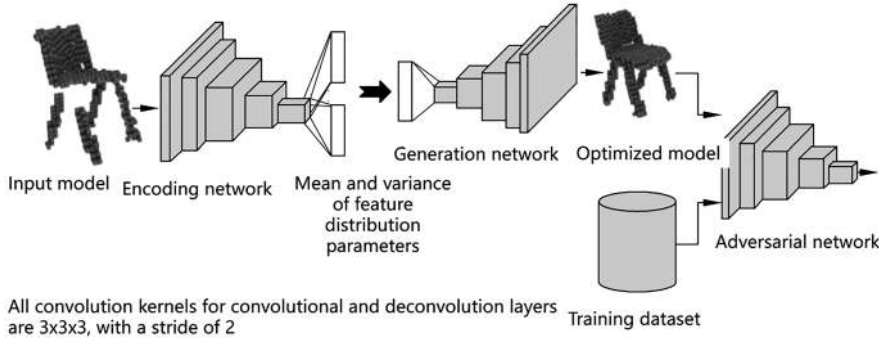


Fig. 1.14 Reconstruction completion optimization module network architecture diagram

training data in database Y into batches to the network B to be trained, and calculate the errors in the reconstruction completion optimization module according to formulas (1.26) to (1.29). Update the network parameters iteratively using the gradient descent optimizer according to the method of backpropagation until the number of iterations meets the requirements, and complete the training of the reconstruction completion optimization module network.

- In Fig. 1.13, only the data flow situation of one-time step is drawn. The network includes the reconstruction module and the viewpoint dynamic prediction module, which are used to receive the reconstruction model and predict the next view.

- The network in Fig. 1.14 includes the encoding network, the generation network, and the adversarial network. Only the encoding network and the generation network need to be run during the inference stage, and the adversarial network is only used during training.
- **Reconstruction Process:** For a object to be reconstructed, manipulate a robot or a handheld RGB-D camera to come to a random position around the object, and ensure that the object exists in the camera's field of view at this angle. Take a photo with the camera, use the RGB three channels in it, and input it into the already trained network after compressing the resolution. At this time, the object reconstruction module will output the currently reconstructed model, and the viewpoint dynamic prediction module will output the next best view. The robot or handheld RGB-D camera comes to the next best view, continues to take pictures, sends them to the already trained network, and obtains the reconstructed model and the next view. This process is repeated until the reconstructed model meets the requirements or the number of views has reached the preset threshold. The output at this time is the object reconstruction model, as shown in Fig. 1.15. The reconstruction starts from a random view, takes pictures under this view, and passes them through the pretrained viewpoint dynamic prediction module and the object reconstruction module. The former is responsible for receiving the picture and the current view and judging the best view. The latter is used to reconstruct the current 3D model. This is a repetitive process, the viewpoint dynamic prediction module repeatedly judges the next view to obtain the next picture input, and the object reconstruction module continuously reconstructs the

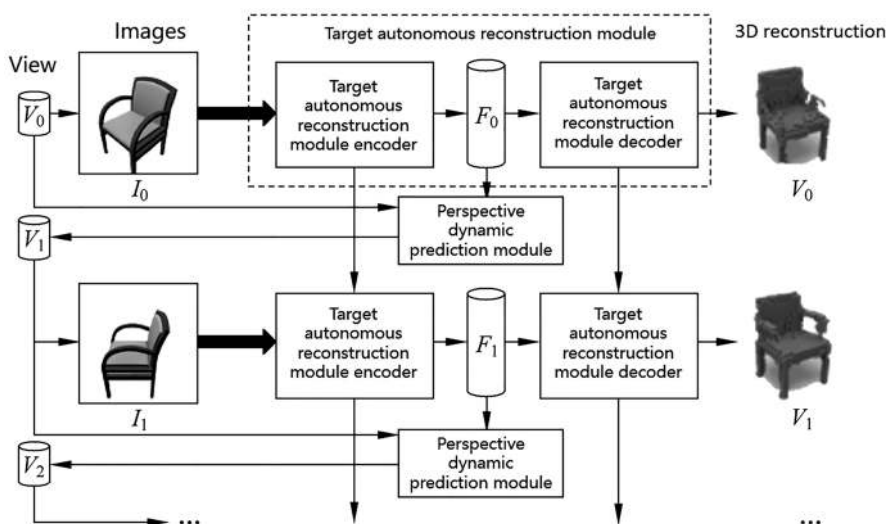


Fig. 1.15 Reconstruction process

model until the iteration ends. Finally, the reconstructed model is input into the generation network of the reconstruction completion optimization module to obtain the final optimized reconstruction model.

- **Reconstruction Completion Optimization Process:** For the reconstruction results obtained in the reconstruction process, input them into the encoding network in Fig. 1.14 to obtain the feature distribution of the reconstruction model, and then input it into the generation network in Fig. 1.14 to obtain the model that has been completed and optimized by the network. The discriminative network is only used when training the network model and is not used at this stage.

1.3.2.4 Results Display

1. Dataset Establishment

In the establishment of the dataset for the training of the active reconstruction process (including the viewpoint dynamic prediction module and the object reconstruction module), since the network input is a sequence of images from different views and the output is the 3D voxel of the object, it is necessary to establish a data pair of the image sequence and the 3D voxel ground truth. The dataset is constructed based on the dataset released by PTN [27], and the CAD models of 13 categories in ShapeNet [24] are used as the objects to be reconstructed to establish the data pair of the image sequence and the 3D voxel ground truth. For each object's 3D CAD model, a 3D body of $32 \times 32 \times 32$ voxels (the orientation of the 3D body is consistent with the standard direction, each voxel occupies that spatial position, 0 means not occupied, 1 means occupied) is used as its 3D shape ground truth; 24 views are selected around the object at a 30° elevation angle, the adjacent views are spaced 15° in azimuth, and the 24 views just cover the object for one round. Under the same set of camera lighting conditions, render the color map of the 3D model at the selected 24 views; the color map contains RGB 3 channels; and the image is finally obtained by center cropping and shape reduction to a height and width of 64×64 pixels. The final generated dataset consists of object categories, object indexes, 3D body ground truth, view sequences, and image sequences.

In the establishment of the dataset for training the reconstruction completion optimization module, the ShapeNet [24] database model is virtually scanned to obtain a single-view depth map of the 3D model, and a deficient point cloud is generated by fusing a single or a small number of depth maps, which is then converted into deficient 3D voxel data. This synthetic dataset method can provide a large amount of deficient data. At the same time, voxelizing the complete 3D model in ShapeNet to obtain data pairs of deficient 3D shapes (3D shapes are represented by 3D bodies, and the voxels inside the 3D bodies represent the grid occupancy probability) and complete 3D shapes.

Since the reconstruction module in this technical case refers to the literature [29] in 2018, it is compared with the multi-view voxel reconstruction method

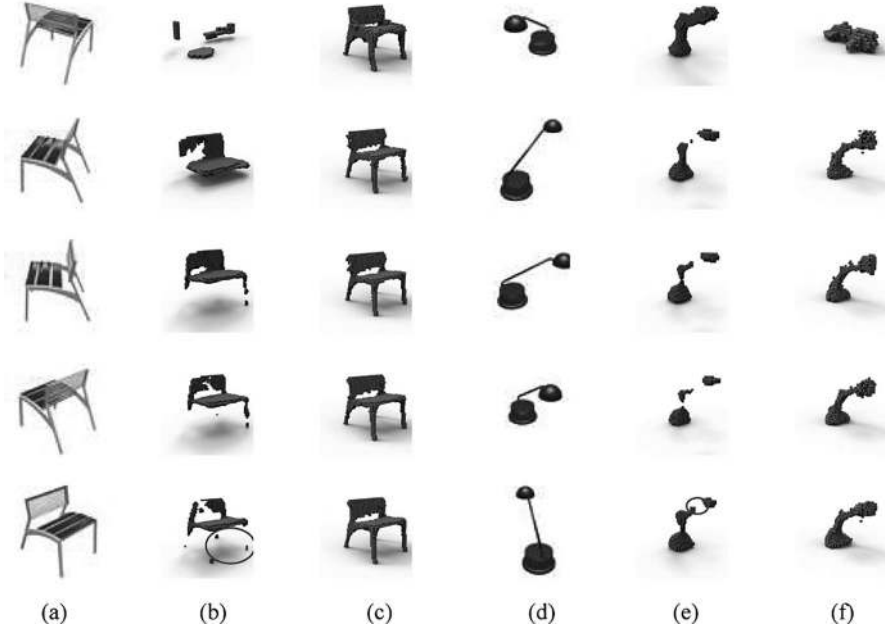


Fig. 1.16 Reconstruction result display

3D-R2N2 [27] close to the time of this literature in the evaluation. 3D-R2N2 uses a 2D encoding-3D recurrent layer-3D decoder structure. Due to the presence of the recurrent layer, this network can not only realize the 3D reconstruction of single-view images but also realize the 3D reconstruction of multi-view images, and it can also verify the reconstruction effect of this technical case system in the aspect of multi-view image 3D reconstruction.

Figure 1.16 shows the reconstruction results under different numbers of views. Figures (a), (b), and (c) in the figure, respectively, represent the different view input images of the chair object (from top to bottom, the number of view inputs increases from 1 to 5), the 3D-R2N2 voxel reconstruction results, and the voxel reconstruction results of this technical case; Figures (d), (e), and (f), respectively, represent the different view input images of the desk lamp object, the 3D-R2N2 voxel reconstruction results, and the voxel reconstruction results of this technical case. From the picture, it can be intuitively seen that the system can even predict the global structure with only one view, and as the number of views increases, it gradually completes and optimizes the local details of the reconstruction results. The optimization of local details can be seen from the part circled in the figure. Compared with 3D-R2N2, the system of this technical case can better use different view information to complete the local structure that was previously ignored in the reconstruction.

2. Completion Optimization Quality Evaluation

Figure 1.17 shows the completion effects of different types of models. Each row represents a type of object model, and the input and corresponding output

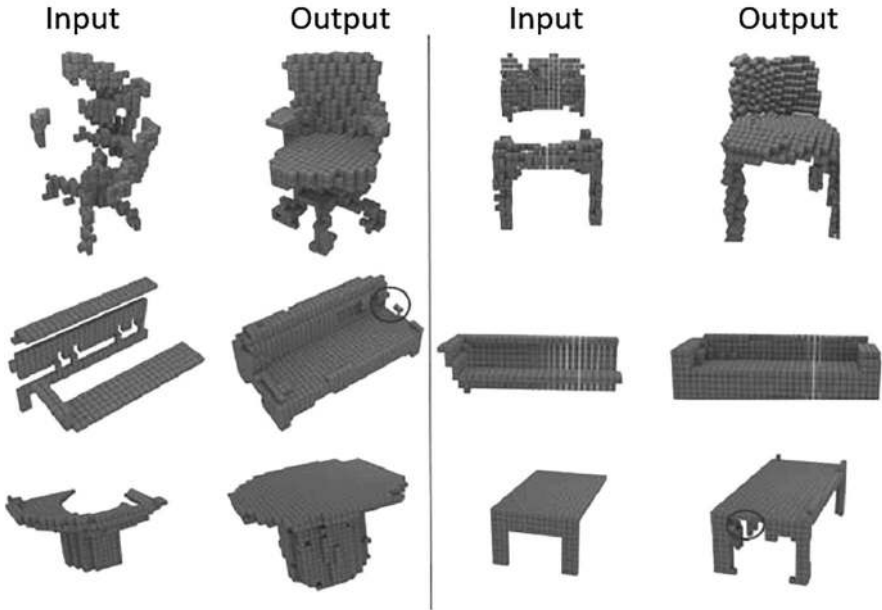


Fig. 1.17 Completion effects of different types of models

effects are shown in different columns. It can be seen that this technical case can restore the complete structure of different types of models. Although the optimization of generative adversarial networks gives it stronger generative capabilities and more detailed output results, it is undeniable that this method still has certain problems, such as difficulty in balancing the quality and authenticity of reconstruction and slow convergence speed of the network. In addition, the generation method in this technical case inevitably has some noise and lack of detail, mainly reflected in the inability to predict the finer parts of the object well. One reason may be that the structure information of the finer parts is lost during the convolution and deconvolution process; the second reason may be that the resolution used is relatively low. Octrees, point clouds, or other memory-saving representation methods can be used to improve resolution or representation ability in detail. The areas with poor effects are shown in the circle in Fig. 1.17.

1.3.2.5 Summary

This technical case starts from the practice of object reconstruction based on deep learning, introduces the voxel modeling method of 3D objects based on dynamic viewpoint planning, and the 3D shape completion optimization method based on the combination of variational autoencoder network and generative adversarial network. By building a viewpoint dynamic prediction module, a object reconstruction

module, and a reconstruction completion optimization module and training the network, it can dynamically plan the scanning viewpoint for the given object, and at the same time combine the pictures under different viewpoints to complete the construction of the 3D object model, and finally complete the optimization for the incomplete parts. The results show that the network can make reasonable structure predictions when at a single viewpoint and can continuously improve the local details of the reconstructed structure as new predicted viewpoints arrive, improving the quality of reconstruction; and in the task of model optimization and completion, it can also make reasonable generation predictions for incomplete models.

However, the method provided in this technical case only extracts semantic features and ignores the geometric relationship between different viewpoint pictures and 3D structures. In addition, the voxel representation method takes up too much memory, resulting in a low output resolution ($32 \times 32 \times 32$), which makes it difficult to express smaller local structures, such as the part connecting the lamp and the lamp base in Fig. 1.16 is too thin, and the quality of reconstruction is not ideal. This problem can be alleviated by using data representations such as octrees and point clouds, as well as more powerful network design and training strategies. The purpose of this technical case is to help everyone understand how to use deep learning methods to reconstruct and optimize object models. Better model design and more problem solutions are left for readers to explore further.

1.3.3 Case Study: Scene Modeling Based on Template Replacement

1.3.3.1 Task Description

3D reconstruction of a scene, as opposed to 3D reconstruction of an object, involves 3D modeling of all visual information in the entire scene. The quality of the 3D scene model has a significant impact on research in fields such as robot interaction, scene understanding, and autonomous driving. In recent years, with the update of 3D acquisition equipment and the improvement of modeling algorithm performance, the quality of modeling has greatly improved. However, due to some objective limitations during the acquisition process, the model often has missing and noisy situations, and the accuracy of the model cannot meet the requirements. To solve this problem, it is necessary to optimize the model obtained by scanning. Currently, there are mainly two methods to solve this problem: the first method is model completion, which is to fit the missing parts using the point cloud information around the missing area; the second method is model replacement, that is, finding the model most similar to the scanned model in the database, and then registering the retrieved database model with the original scanned scene. This technical case adopts the latter method of model replacement. Its input is the point cloud scene model with missing areas directly collected by the sensor. By replacing the

incomplete objects in it with similar models in the database, the final output is a complete scene model.



Chapter Knowledge Points

Point Cloud: A data set of points in space that can represent 3D shapes or objects. The position of each point in the point cloud is described by a set of Cartesian coordinates (X, Y, Z), some of which may contain color information (R, G, B) or object reflection intensity information. The acquisition of intensity information is the echo intensity collected by the laser scanner receiver, which is related to the surface material, roughness, incident angle direction of the target, and the emission energy and laser wavelength of the instrument. The point cloud is also a set of point data measured by instruments in reverse engineering.

Voxel: Short for volume pixel. Conceptually similar to the smallest unit in 2D space—pixels, which are used in 2D computer image data. As the name suggests, a volume pixel is the smallest unit of digital data in 3D space division, used in 3D imaging, scientific data, and medical imaging, etc. Some true 3D displays use voxels to describe their resolution, for example, a display that can display $512 \times 512 \times 512$ voxels.

Polygon Mesh: In 3D computer graphics and solid modeling, a polygon mesh is a set of vertices, edges, and faces that define the shape of a polyhedral object. The shape of the face is usually represented as triangles, quadrilaterals, or other simple convex polygons. Its data structure is simple, can retain the texture, geometry, and other information of the model, and can express various irregular objects well. In the computer world, because only discrete structures can be used to simulate continuous things in reality, the surfaces in the real world are actually composed of countless small polygonal faces in the computer.

Iterative Closest Point (ICP) Algorithm: An algorithm used to register two or more groups of point clouds under different reference coordinates based on the consistency of object geometric information. The basic idea is to find the nearest points in the target point cloud and the source point cloud to be matched, according to certain constraints, and then calculate the optimal matching matrix to achieve the purpose of minimizing the error function.

1.3.3.2 Related Work

In recent years, more and more shared indoor scene 3D model libraries have appeared on the Internet. Existing data-driven indoor scene research can be roughly divided into two categories: object recognition, reconstruction in the scene, and scene object layout optimization. In terms of object recognition and modeling of scanned point clouds in indoor scenes, scholars in our country have done a lot of excellent work. Hu Shimin and others from Tsinghua University [63] proposed an interactive reconstruction algorithm, which first collects user data. RGB-D images are automatically segmented, and

users can interactively modify the segmentation results. Then, by matching the segmented object point cloud with the database model, 3D modeling is completed. Nan Liangliang et al. [60] proposed to use the model database to assist incomplete point cloud data, segment and identify the point cloud in the scene by searching and classifying, and complete the point cloud through template fitting, thereby obtaining the 3D model of indoor furniture. Both fully utilize the advantages of big data and effectively solve the problems of large noise and incompleteness of scanning point clouds. If there is only a single 2D image, the 3D furniture model library with semantic annotation can also be fully utilized. By matching the model and image content, the geometric estimation and furniture recognition of the scene image can be realized, such as the modeling algorithm proposed by Fouhey et al. [64]. However, due to the inherent limitations of the image, this algorithm depends on the accuracy of the camera parameter estimation. When the camera parameter estimation is inaccurate, it is impossible to obtain accurate scene reconstruction results just by using the database.

1.3.3.3 Technical Method

The technical process includes preprocessing, 3D model voxelization, feature and characteristic extraction, and scene registration, as shown in Fig. 1.18, specifically as follows.

1. **Preprocessing:** Adopting the single-view modeling framework proposed by DAI et al. [13] for initial modeling of indoor scenes, this algorithm proposes a novel, real-time, end-to-end reconstruction framework to complete the modeling task, improving the original pose estimation and frame optimization strategy, and adopting an efficient hierarchical method based on color images and depth maps as inputs. Since the modeling framework proposed in this technical case retrieves, replaces, and registers models in the database for individual target objects, it is necessary to segment the original scene to obtain the target object with missing information. Therefore, it is planned to use the point cloud semantic segmentation algorithm proposed by Qi et al. [65] to segment the scene model and segment the incomplete object point cloud. This method uses point symmetry functions and symmetry networks to solve the problems of nonfixed point cloud order and point cloud rotation and introduces new loss items to constrain

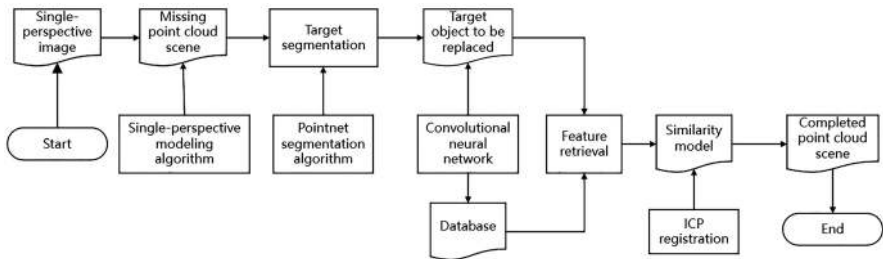


Fig. 1.18 Scene modeling process based on template replacement

the orthogonal situation caused by a large number of parameters. Finally, the network uses a multilayer perceptron to generate high-dimensional feature vectors for each point, which are processed by a nonlinear classifier, outputting the classification vector of each point, to complete the segmentation of the scene.

2. **3D Model Voxelization:** To meet the requirements of CNN input, it is necessary to preprocess the target object after segmentation and the model in the database to meet the requirements of regularized data. This includes operations such as extracting the surface normal vector of the object and voxelizing the object point cloud with an octree grid. The first is the extraction of the normal vector. It is planned to use virtual scanning technology to sample the dense area of the point cloud, select the point with the largest change in the normal vector direction as the feature point of the sampling point, and use the normal vector and curvature information of this point as the underlying feature of the point cloud area. At the same time, 14 virtual cameras are placed at the center of the point cloud truncated sphere, facing different directions, and 16,000 parallel rays are emitted in each direction. When the ray intersects with the surface of the point cloud, the intersection point can be regarded as a sample of the surface point. The point with the largest change in the normal vector in the surrounding area of the sampling point is selected as the feature point of this area, and the arithmetic average of the angle between the normal vector of a point and its neighboring points is calculated as the surface normal vector required by the algorithm. Next is the construction of the octree structure. The octree has a 3D grid structure with adaptive size, which is an extension of the 2D quadtree structure in 3D space. Compared with the traditional voxel grid, it can significantly reduce the consumption of storage at the same resolution. In the traditional octree structure, each node contains pointers to child nodes, making the time to access a node linearly proportional to the depth of the tree, ultimately leading to a large amount of time required to process operations between nodes, especially when dealing with high-resolution objects. In this technical case, in the modeling framework, the target objects in the scene and the database objects are expressed in the form of an octree. First, the point cloud model is placed in a bounding box of unit length and subjected to a breadth-first recursive operation. The recursive process is as follows: When traversing to the i -th layer of the octree, recursively visit all nodes containing the model boundary and divide the bounding box containing the model boundary into eight equal parts as the $i + 1$ layer child nodes of this node. If a node does not contain any part of the model, stop dividing this node.
3. **Feature Extraction:** In order to abstract more generalized features of 3D objects, this technical case includes an octree grid with normal vector and curvature information as input, and the output high-dimensional feature vector serves as the basis for database retrieval. The goal of the feature extraction network is to map the input point cloud model to a vector in high-dimensional space. For the octree grid that can separately input curvature information and normal vector information, first perform 4 convolutions and downsampling operations on the above 2 grids to obtain two 64-dimensional feature vectors. Next, the two features are fused and input into the subsequent feature extraction network. After 2 convolutions and downsampling operations, a 256-dimensional vector is obtained. Finally, the 256-dimensional feature vector is input into the fully con-

nected layer, and a 55-dimensional vector is obtained after the normalization exponential function operation. This vector is the output result of the feature extraction network. This technical case performs batch normalization on the data after the convolution operation, which can eliminate the impact of data distribution on network training, and adds a nonlinear activation function, namely, the rectified linear unit activation function, before the downsampling operation, thereby improving the model's expressive power. To prevent overfitting, this technical case adds a Dropout layer after the fully connected layer.

4. **Scene Registration:** Use the Iterative Closest Point [66] algorithm to register the retrieved database objects with the initial scene to complete the final modeling process. Its algorithm is a registration algorithm aimed at achieving the optimal least squares method, which is a common method for aligning point clouds. This algorithm continuously calculates the corresponding rotation and translation matrices by repeatedly selecting corresponding point pairs in adjacent relationships and repeatedly repeats this process to reduce the gap between the two point clouds and finally complete the registration process. The above process can be described as inputting two point cloud scenes under different coordinate systems, finding the spatial transformation matrix between the two scenes, and completing the registration process. The key to this method is how to find corresponding point pairs in the scene and how to calculate the transformation matrix based on the corresponding point pairs to minimize the distance between point clouds. In order to overcome its sensitivity to the initial position of the point cloud scene, especially when the initial transformation matrix of the point cloud is chosen extremely unreasonably, leading to the problem of falling into local optimal points. This technical case makes the scanning model and the database model face the same direction in the preprocessing stage, obtains the corresponding pose parameters by registering the target object with the database object, and finally completes the process of scene modeling through fusion.



Sample Code

Input: Single-view image

Output: Scene model after template replacement

1. Use the BundleFusion method to perform initial modeling on the input image;
2. Use the PointNet method to perform semantic segmentation on the model;
3. FOR object model in [set of objects after segmentation];
4. Voxelize the object model;
5. Inputting the object voxel into the convolutional neural network for feature extraction;
6. Retrieving the most similar object model from the database based on the feature vector;
7. Using the ICP algorithm to register the model in the database;
8. Restoring the registered model to the initial scene model;
9. END.

1.3.3.4 Results Display

After obtaining the corresponding features through the feature extraction network, the object to be replaced is matched and retrieved with features from the database model to obtain the most similar model set. As shown in Fig. 1.19, for the original incomplete chair (the far left in the figure), the five most similar chairs are retrieved from the database, with the similarity decreasing from left to right.

The specific steps of model registration are as follows: First, keep the positive direction of the two models consistent, and then continuously match the two point sets through the nearest point matching algorithm to complete the scene registration task. The registration result is shown in Fig. 1.20, where the chairs and tables in the scene are replaced by the most similar models in the database.

1.3.3.5 Summary

In summary, the indoor scene modeling framework based on template replacement proposed in this technical case is to use similar 3D models in the database to replace the target objects with missing information, and then complete the scene modeling task through scene registration. This framework has good modeling effects when there are severe structural losses in the scene and a large part of the target objects



Fig. 1.19 Database chair retrieval example

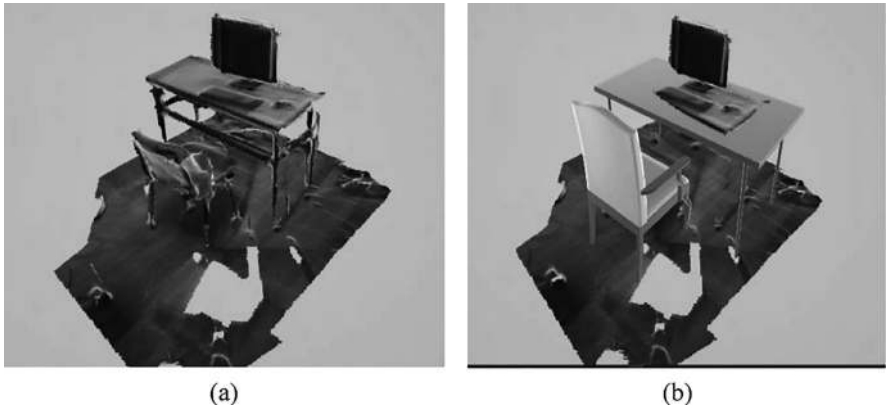


Fig. 1.20 Scene modeling result indication. (a) Initial modeling scene; (b) Replacement scene

are missing. The reason is that the template replacement method uses the whole replacement method for modeling. As long as the feature extraction network can get good features and can retrieve similar models, the scene restoration effect is often obvious. Therefore, when the information in the scene is severely missing, the modeling method of template replacement often has good effects. However, when the task to be completed requires more detailed modeling effects, or when the requirements for realism are high or sensitive to position information, due to the limitation of the database capacity, the modeling framework based on template replacement will cause the retrieved model to have a large deviation from the real object. At the same time, because this technical case uses scene registration for reconstruction, the error of registration will also affect the quality of the model. When the error of registration is large, it will cause a large deviation in the position and pose of the 3D object and not get satisfactory results.

1.3.4 Case Study: Scene Reconstruction Based on Unsupervised Learning

1.3.4.1 Task Description

In this technical case, unsupervised learning is used to reconstruct the 3D scene from continuous video frames, and the scene model is represented by point clouds. Specifically, sufficient texture color information and geometric information can be obtained from the color images in the video and the predicted depth images, realizing the reconstruction of the scene point cloud, and the 3D scene point cloud can be reprojected to the 2D video frames, and then use their consistency to correct the scene point cloud, realizing the relatively complete reconstruction of the scene point cloud under unsupervised conditions.



Chapter Knowledge Points

Camera Extrinsic Matrix: Indicates the position of the camera when a certain image was taken and the angle relative to different axes of the world coordinate system, often represented as a (R, t) matrix. The R matrix is a rotation matrix, which changes the direction of a vector when multiplied by it but does not change its magnitude and maintains the chirality of the matrix.

Depth Rendering: When a 3D point cloud model and a certain camera pose are given, microscopically, according to the pose matrix, each point in the point cloud model can be projected to the image plane coordinate system according to its 3D coordinate information, and the value of this pixel represents the distance from the 3D point to the camera. Macroscopically, it is equivalent to taking a photo of the 3D model at a specific location with the camera, but the photo stores depth information.

Pinhole Camera Model: The process of a camera mapping coordinate points in the 3D world to a 2D image plane can be described with a geometric model. There are many types of this model, the simplest of which is called the pinhole model. In real life, a pinhole camera is made up of a small hole (pin-hole) in the front. Light from an object in the real world passes through this hole and forms an inverted image on the camera's baseplate or image plane.

Camera Intrinsic Matrix: Under the pinhole model, if the projection center is used as the origin of the coordinate system, and the X-Y coordinate axis plane is parallel to the image plane to build the camera coordinate system, any point in space can be mapped to a point on the image plane. In this process, a matrix composed of the coordinate axis scale factor (i.e., the pixel of the unit length inside the camera), skew factor, and the coordinates of the principal point on the image plane is formed. This matrix is called the camera intrinsic matrix, which reflects the correspondence between points in space and points on the image plane.

1.3.4.2 Related Work

In recent years, the PointNet series of works proposed by SU [65, 67], Li [68], etc., have provided many references for the feature extraction problem of point clouds. In recent years, literature [69–71] used depth maps or RGB maps to complete the reconstruction of scene voxel grids, compared to point cloud representation. On the one hand, voxels limit their resolution, and when using 3D convolutional neural networks, they require a huge computational cost. These works also need the voxel scene of GT as supervision, and the 3D representation of the real scene is difficult to obtain. On the other hand, without the camera pose, compared to the depth map, the direct connection between the RGB color map and the 3D scene model is more difficult. Therefore, it is considered to predict its depth map and camera pose as an intermediate medium before synthesizing the scene point cloud, so as to more easily complete the scene reconstruction. The video sequence well satisfies these two conditions. Inspired by literature [72–74], the consistency relationship between adjacent frames in the video can be used as a supervisory signal to learn the relative relationship of the camera pose from the monocular video sequence and predict the depth map corresponding to each frame through the prediction of the camera pose between the previous and next frames.

1.3.4.3 Technical Method

As shown in Fig. 1.21, the first part is the depth map and camera pose prediction module. The blue part in the figure is the depth map prediction network, and the green part is the camera pose prediction network. By learning the relationship between adjacent frames in the video, the depth map and its camera pose

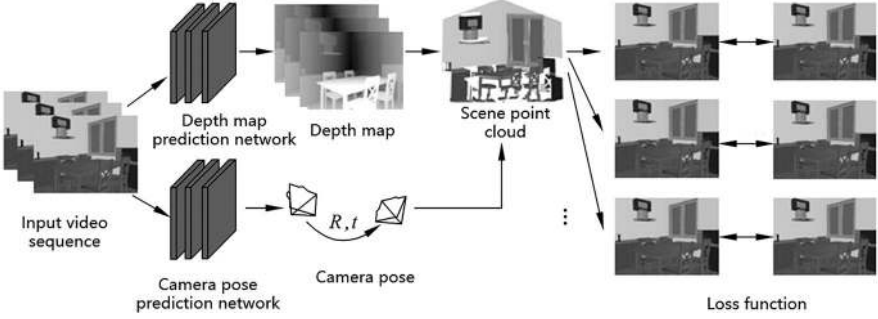


Fig. 1.21 Unsupervised learning-based scene reconstruction (see color figure before the text)

corresponding to each frame are predicted. The depth map prediction network consists of an encoder and decoder with multi-scale prediction outputs with skip connections. The encoder and decoder each contain seven layers of convolution or deconvolution. The first two layers of the encoder's convolution kernels are 7×7 and 5×5 , respectively, and the remaining convolution layers all have 3×3 kernels. Each convolution is followed by a ReLU activation layer, and the output depth map size is the same as the input video sequence size. The camera pose prediction network consists of 7 convolutions with a stride of 2, each followed by a 1×1 convolution, with 6 output channels 3 for translation angles and 3 for rotation angles, representing the camera pose. Finally, an average pooling layer is applied to aggregate the camera pose predictions at all spatial locations. The multi-task joint training model is implemented through a Meta-Critic network. The Meta-Critic network consists of two modules, the MVN (Meta-Value Net) module and the TAE (Task-Actor Encoder) module. The input to the TAEN includes the input sequences of the DepthActor and CameraActor, the predicted camera pose and depth map sequences, and the difference between the predicted sequence and the ground truth. Its function is to encode this useful information into a feature vector z ; Specifically, the TAEN is defined as a recurrent neural network with LSTM, ending with a fully connected layer that maps the distributed features of the last LSTM timestep to the feature vector z . The input to the MVN includes the input sequences of the DepthActor and CameraActor, the predicted camera pose and depth map sequences, and the feature vector z from the TAEN. Its function is to determine the accuracy of the predictor Actor network and use this as supervision to continuously adjust the parameters of the two source task Actor networks and the Meta-Critic network during the training process through neural network learning, thereby improving the learning ability of the Meta-Critic network.

The optimization objective of the Meta-Critic network is represented as

$$\theta^{(i)} \leftarrow \operatorname{argmax}_{\theta^{(i)}} Q_{\phi} \left(x^{(i)}, \hat{y}^{(i)}, z^{(i)} \right) \forall i \in [1, 2, \dots, M] \quad (1.30)$$

$$\phi, \omega \leftarrow \operatorname{argmin}_{\phi, \omega} \sum_{i=1}^M \left(Q_{\phi} \left(x^{(i)}, \hat{y}^{(i)}, z^{(i)} \right) - r^{(i)} \right)^2 \quad (1.31)$$

where x is the input value, y is the predicted value, z is the feature vector encoded by TAEN, r is the difference between the input and the ground truth, and θ , φ , and ω are the network parameters of the task predictor, MVN, and TAEN, respectively. From eqs. (1.30) and (1.31), for each task M , the task predictor Actor network will learn to maximize the value function during the training process, making the prediction results more accurate. Compared to traditional supervised learning that minimizes a fixed loss function, this is more universal. At the same time, for each task, the Meta-Critic network will learn to simulate the actual supervised learning loss function. For single-task problems, this undoubtedly increases the workload, but for the multi-task joint training of this patent, it can share some prior knowledge of task crossover.

The second part is the scene point cloud synthesis module, which combines the predicted depth map sequence with the initial RGB video sequence and the predicted camera pose information for fusion calculation to obtain the fused 3D point cloud model. The target task is to reconstruct the point cloud model of the 3D scene from the video; The target task model includes a point cloud prediction module and a Meta-Critic module, the Meta-Critic module is the same as the multi-task joint training model. The input to the point cloud prediction module is a continuous color video sequence, and the output is a 3D point cloud. The network is built based on standard LSTM and GRU modules. The GRU module retains the effective features of the previous sequence, which is used to fine-tune the scene point cloud reconstructed based on the previous sequence features, making the scene point cloud contain more local and global detail features. The scene point cloud reconstruction model uses the VGG structure and employs a fully convolutional network, and all convolutional layers use 3×3 convolution, with ReLU nonlinear activation layers. Through continuous downsampling, a 3D point cloud model is regressed.

The third part is the point cloud re-projection network module, which re-projects the fused color point cloud to the 2D plane according to the camera pose's viewpoint, obtaining a 2D RGB image. The re-projected RGB image sequence is compared with the original video frames for network optimization, achieving gradual optimization and improvement of the 3D point cloud model. The experimental process is divided into three stages—multi-task joint training stage, target task training stage, and target task testing stage. The multi-task joint training stage is trained on the 7Scenes dataset. During each training process, multiple consecutive images are fed into the multi-task predictor Actor network as sampling inputs, and the output predicted depth maps and camera pose results are sent to the Meta-Critic network for supervised training. The network parameter model is optimized in continuous iterative training until the model converges. The target task training stage combines the 7Scenes dataset and the dataset collected from SUNCG for training. In this stage, the parameters of the Meta-Critic network of the model trained by multi-task joint training remain unchanged, and the scene point cloud reconstruction model predictor Actor network is trained to optimize the scene point cloud reconstruction model. In the target task testing stage, only the scene point cloud reconstruction model optimized in the target task training stage is used, inputting continuous video sequences, and outputting 3D scene point clouds.

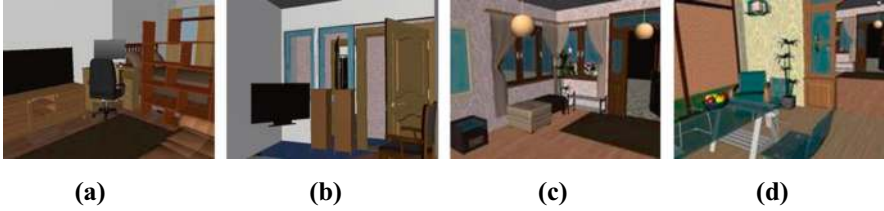


Fig. 1.22 Single frame color image

Sample Code

Input: Video sequence

Output: Scene point cloud model

1. FOR RGB image in [video sequence];
2. Input the RGB image into the depth prediction network to get the depth image;
3. Input the RGB image into the camera pose prediction network to get the pose of each frame;
4. Re-project the predicted depth map into a point cloud according to the predicted pose information, and attach the corresponding RGB information;
5. END;
6. Merge all the above point clouds into a complete scene point cloud model according to the coordinate information.

1.3.4.4 Result Display

Since the 3D scene reconstruction uses unsupervised learning methods, and the method also considers the difficulty of obtaining 3D scene point clouds, the experimental results are mainly qualitatively analyzed. The training and testing datasets of the scene model focus on common indoor scene layouts, such as bedrooms, living rooms, and office desks and chairs. For each scene point cloud, 43,200 3D points are used for description and display, which will not cause a huge network load.

The experimental results show a color image in the video sequence and the scene point cloud generated by the network. The color image is shown in Fig. 1.22, and its corresponding scene point cloud is shown in Fig. 1.23.

1.3.4.5 Summary

In response to the difficulty of obtaining 3D supervised data and the nonideal results of existing methods, this technical case uses unsupervised learning to complete the point cloud-level indoor scene reconstruction work. By predicting the depth and

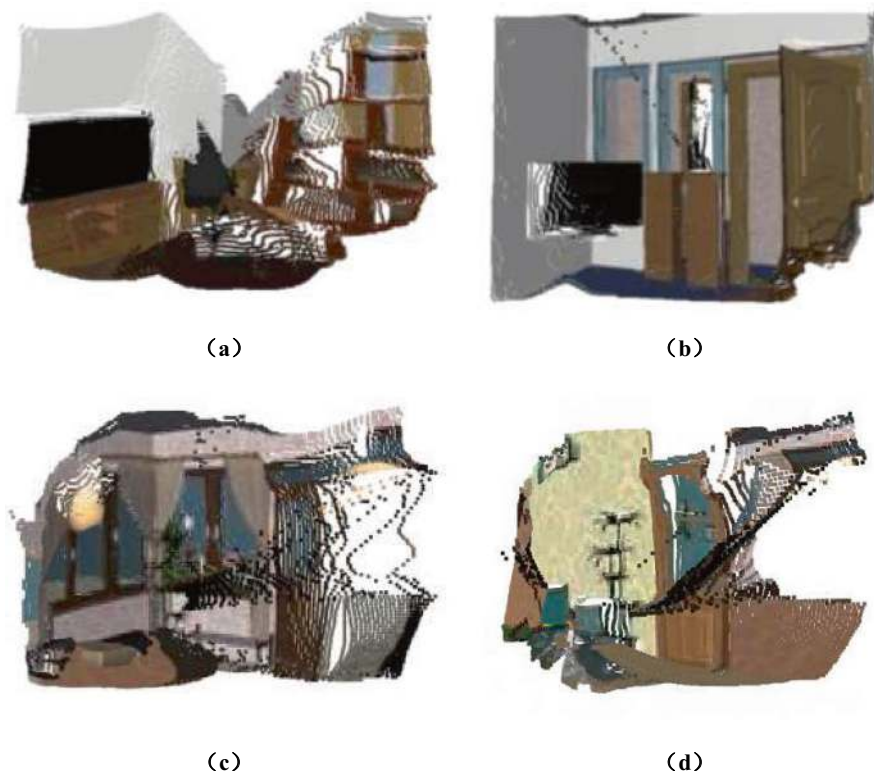


Fig. 1.23 Scene point cloud reconstruction result

estimating the camera pose of the input video sequence, the conversion from the video sequence to the point cloud model is realized. At the same time, according to the predicted camera pose, the depth information of different frames can be constrained for consistency, thereby ensuring the training of the network model under the premise of unsupervised information.

1.3.5 Case Study: Scene Reconstruction Based on Single View Image

1.3.5.1 Task Description

The problem of 3D modeling based on images, as a classic problem in computer vision, has been studied by a large number of scholars for a long time. Traditional methods are mostly based on inputting multiple images from adjacent viewpoints, using inter-frame matching or feature point matching relationships to integrate

images from multiple viewpoints into a whole, and outputting an optimized 3D model. The task of this technical case is to extract sufficient structural feature information from a single RGB-D image through deep learning, and finally reconstruct a complete textured point cloud scene model. For a given single RGB-D image, if it is directly back-projected into a 3D point cloud representation, due to the limitations of a single viewpoint, this point cloud will have a lot of missing information. The goal of this technical case is to complete these missing point clouds and thus complete the reconstruction of the scene.



Chapter Knowledge Points

Image Inpainting: Image inpainting is a problem between image editing and image generation, and it was originally a traditional graphics problem. The problem itself is very intuitive: Dig a hole in an image, how to use other information to fill this hole, and make it indistinguishable to the human eye. Its purpose is to require the algorithm to complete the missing area of the image to be repaired based on the image itself or image library information, so that the repaired image looks very natural and is difficult to distinguish from the undamaged image area.

Point Cloud Completion: Due to the limitations of the accuracy and resolution of 3D scanning equipment, the scanned 3D models often have defects or noise. In order to better restore the appearance of the real world object or scene itself, it is necessary to repair the incomplete point cloud model, that is, to predict some reasonable 3D points to complete the model, ensuring the completeness and unity of the object or scene.

Semantic Scene Completion (SSC): In order to perform advanced tasks such as navigation, interaction, or object retrieval, robots need to have the ability to understand the surrounding environment at a semantic level. Semantic scene completion is a computer vision task that combines 3D shape completion and 3D semantic segmentation, which can help robots perceive the 3D world and interact with the environment. Its goal is to complete semantic segmentation while completing the shape of the scene.

Multi-view Convolutional Neural Network (Multi-view CNN): The input of this network is multi-view image information. After convolution, the deep features of the object are extracted through the operation of maximum pooling, which is used for subsequent object recognition, classification, and other tasks. Specifically, the 2D rendering images obtained from different “views” of the object's 3D data are used as the original training data, and the classic and mature 2D image convolution network is used for training. The recognition and classification effects of 3D objects are better than the models trained directly from 3D data.

1.3.5.2 Related Work

Song et al. [70] first proposed a 3D end-to-end convolutional neural network SSCNet, which takes a single depth map as input and outputs a scene voxel model with semantic segmentation information from the corresponding view. Dai et al. [75] expanded SSCNet to solve larger scenes with spatial variations. Wang et al. [76] added an adversarial network structure to SSCNet, making the output results more realistic. Zhang et al. [69] modified SSCNet, using a dense CRF model, further improving the accuracy of the output results. In order to better mine the information of the input image, Garbade et al. [77] used a dual-stream neural network to maximize the extraction of depth information and semantic segmentation features from the input RGB image. Guo et al. [78] proposed VVNet, which can extract structural feature information from depth maps and pass this information to 3D voxels, thereby achieving better scene reconstruction performance. However, the above works are all based on voxel representation, and the resolution of their output results is relatively low. This technical case intends to complete the depth image sequence with camera pose information, directly predict the missing point cloud area at the point cloud level, and thus obtain a complete indoor scene model with a higher resolution expressed as a point cloud.

1.3.5.3 Technical Method

This technical case can initially reconstruct the 3D point cloud model of the indoor local scene from as few views as possible. The core idea is to decide the projection view through reinforcement learning, project the 3D point cloud to two dimensions under this view, and perform depth image completion operations in 2D space, thereby directly predicting the missing 3D information of the model. Taking depth image D_0 as input, it is first converted into point cloud P_0 , and the data of this point cloud P_0 is severely lost. The purpose of this technical case is to generate 3D points to complete P_0 . The key point of the algorithm is to represent the incomplete point cloud as a depth map under multiple views, perform a 2D completion task, and then reproject the completed 2D points onto the 3D incomplete point cloud. In order to make full use of the context information, perform completion operations on each view in an accumulative manner, retain the predicted points generated by the current view, and use them to help complete the next view. This technical case completes the depth map through the voxel-guided view completion module, selects the best view path through the progressive scene completion module, and finally obtains the completed scene point cloud.

As shown in Fig. 1.24, the network structure consists of the following modules:

1. **Projection Module:** The input of this module is the RGB image taken by the robot camera and the predicted depth and semantic segmentation images. First, the depth map is reprojected into 3D point cloud data according to the camera parameters at the time of image shooting, and then the RGB color information

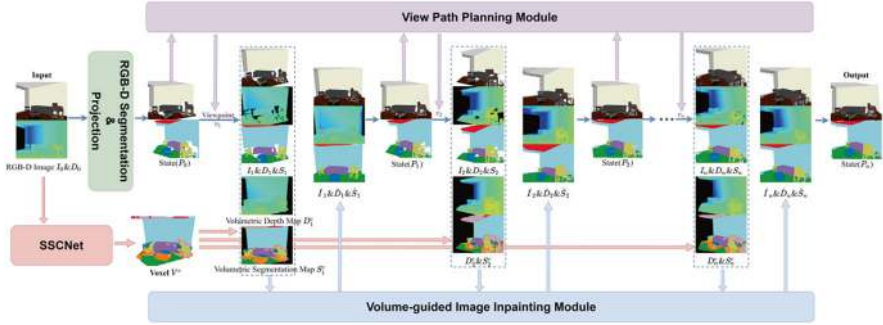


Fig. 1.24 Network structure diagram

and semantic information are added to the corresponding point cloud according to the pixel coordinates.

2. **View Planning Module:** The input of this module is the initial or point cloud scene model during the completion process. Its network architecture is based on the multi-view convolutional neural network (MVCNN). The reinforcement learning algorithm of this module uses the A3C algorithm, which projects the input point cloud model under 20 regular camera views and decides the next best completion view.
3. **Image Completion Module:** The input of this module is incomplete depth images, RGB images, and semantic segmentation images. Through the guidance of voxel depth maps and voxel semantic maps, the missing areas of the above incomplete images are completed. The completion process of depth images and RGB images needs the help of semantic segmentation images.

The specific scene completion steps are as follows:

- (a) **Initial Incomplete Point Cloud Generation:** Based on the given 2D depth map D_0 and its corresponding viewpoint v_0 , the 2D depth map D_0 is converted into a 3D point cloud under the camera coordinate system through projection, and the point cloud is transformed from the camera coordinate system to the world coordinate system, finally obtaining the initial incomplete point cloud P_0 in the world coordinate system.
- (b) **View Path Planning:** View path planning is to obtain the optimal next viewpoint sequence v_i ($i = 1, 2, \dots, n$). This problem is defined as a Markov Decision Process (MDP), composed of states, actions, rewards, and agents taking actions in the decision-making process. The state is defined as the point cloud updated after each iteration, denoted as P_i (i represents the number of state updates, $i = 1, 2, \dots, n$), which is accumulated from all previous iteration updates, and the initial state is P_0 . The action space is a fixed set of viewpoints, containing 20 different viewpoints. Specifically, first put P_0 in its bounding sphere and keep it upright; then create two circular paths with the equator and 45° latitude. Choose 20 camera viewpoints evenly on these two paths, with 10 on each circle. All viewpoints face the center of the

bounding sphere. This group of 20 viewpoints is denoted as $C = \{c_1, c_2, \dots, c_{20}\}$. The reward function mainly includes two parts: the hole-filling reward function R_i^{hole} and the image completion reward function R_i^{acc} , encouraging the agent to choose viewpoints that are beneficial for hole filling and have high image completion quality. The agent uses the Deep Q Network (DQN) in deep reinforcement learning, inputs the current state, outputs the corresponding optimal action, and obtains the most rewards from the environment. The specific process is to project the point cloud P_{i-1} ($i = 1, 2, \dots, n$), the projection viewpoint is in the above action space, thus obtaining 20 depth maps formed by projection. Input these 20 depth maps into the agent, calculate according to the designed reward function, and you can get the optimal next viewpoint vi and the corresponding depth map D_i .

- (c) **Voxel-Guided Image Completion:** The voxel-guided view completion framework first completes in the voxel space, converting the volume occupancy grid V of P_0 into a complete version V_c , the specific implementation network uses ForkNet; then projects the voxel V_c , under the best viewpoint vi selected in step 2, to get the voxel depth map D_i^c ; the image completion network chooses PartConv, with the depth map D_i corresponding to the best viewpoint vi selected in step 2 and the voxel depth map D_i^c as input, to generate the completed depth map $\hat{D}_i$.
- (d) **Projecting 2D Points to 3D, Generating Stage Point Cloud:** Under the best viewpoint vi selected in step 2, project the completed depth map $\hat{D}_i$ in step 3 to generate a new 3D point cloud P_i . Aggregating the old point cloud P_{i-1} and the new projected 3D point cloud can generate a more informative, dense completed point cloud P_i . This point cloud P_i can be used as the input for the next stage.
- (e) **Progressive scene completion:** Repeat steps 2–4 until the termination condition is reached. The termination condition is set as $\text{Area}^h(P_i)/\text{Area}^h(P_0) < 5\%$, where $\text{Area}^h(P)$ represents the projection of point cloud P from all camera views in the action space, and the sum of the number of pixels inside the hole in each rendered depth map is calculated, which is the measurement of that area. Through the above operations, the scene point cloud can be progressively completed, and the final output is the completed point cloud P_n .

1.3.5.4 Results Display

As shown in Fig. 1.25 and Table 1.1, to verify the effectiveness of the designed scene reconstruction algorithm, this method has been validated on the virtual synthetic dataset 3D-Front and real scenes. On the synthetic dataset, this method measures the chamfer distance of the point cloud quality. The CD (Chamfer Distance) metric, which is used to measure the distance between the predicted point cloud and the true point cloud (the smaller the CD, the more accurate the result), is significantly superior to other related works that use voxels as the output form. At the same time, it also outperforms other methods on the Fr metric, which measures the

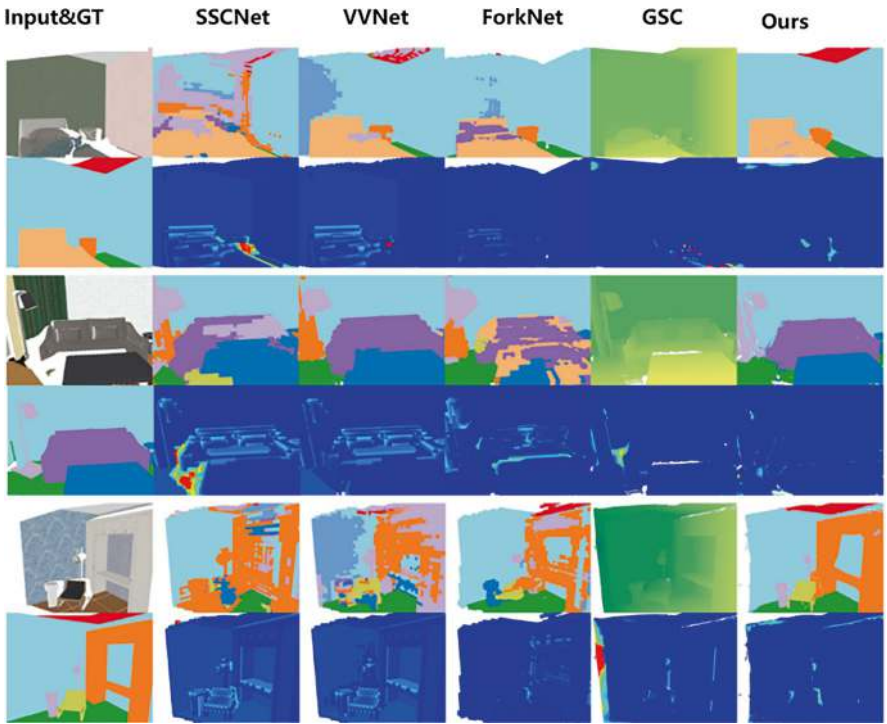


Fig. 1.25 Display of scene reconstruction results on 3D-Front Dataset

Table 1.1 Comparison of results with related work on the 3D-Front Dataset

	RGB	Seg.	RL	CD	$F_s/\%$				
					$F_{r=0.02}$ (%)	$F_{r=0.04}$ (%)	$F_{r=0.06}$ (%)	$F_{r=0.08}$ (%)	$F_{r=0.10}$ (%)
SSCNet	×	√	–	2.1932	71.44	75.28	78.03	79.87	80.95
VVNet	×	√	–	2.1653	75.22	78.31	79.53	80.86	85.08
ForkNet	×	√	–	2.0648	75.86	79.18	80.47	81.81	82.37
GSC	×	×	DQN	0.4897	77.25	81.88	84.28	86.82	87.68
CSC	√	×	DQN	0.4607	74.77	79.58	82.76	84.34	86.69
CSSC	√	√	DQN	0.4328	78.34	83.95	85.53	87.62	88.34
Ours	√	√	A3C	0.4126	78.63	83.57	86.88	88.75	89.42

completeness of the point cloud (indicating the proportion of accurately completed parts in the overall recovery of 3D points, with the center of the sphere being the reconstructed point and the radius being r , the larger the F_r , the more accurate the result). Through experimental comparison, the scene semantic completion network in this technical case has the best scene completion and semantic segmentation accuracy, demonstrating that high-resolution point clouds can achieve more refined

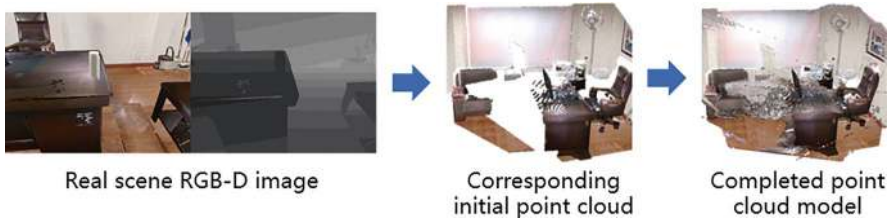


Fig. 1.26 Display of real scene reconstruction results

effects than voxels, and verifying the effectiveness of semantic and 3D geometric constraints for semantic scene reconstruction. For semantic segmentation verification, excluding empty categories, this technical case uses the same semantic classification as SSCNet, including ceiling, floor, wall, window, chair, bed, sofa, table, TV, furniture, and other items. As shown in Fig. 1.26, this method also achieved reasonable reconstruction results in real scenes.

Example Code

Input: Single-view RGB-D image

Output: Scene point cloud model with color and semantics

1. Input the RGB-D image into the semantic segmentation network to get the semantic segmentation image;
2. Reproject the depth image into a point cloud and attach the color information from the RGB image and the semantic information from the semantic segmentation image to the points;
3. WHILE model completion is not finished;
4. Input the point cloud model into the viewpoint planning module to predict the next best completion viewpoint;
5. Project the point cloud under this viewpoint to obtain incomplete depth, semantic, and RGB images;
6. Input the above three images into the image completion module for completion;
7. Reproject the completed depth information into a point cloud, and attach the predicted color and semantic information;
8. Merge the point cloud from step 7 into the initial point cloud model;
9. END.

1.3.5.5 Summary

The current image-based 3D scene reconstruction methods can be summarized as big data-driven depth learning-based mapping methods from incomplete data to complete data. However, these methods have serious problems because they only

support low-resolution data representation: First, when point cloud data is used as the input of 3DCNN, it is downsampled to low-resolution voxels, resulting in serious loss of effective information; Second, the low-resolution representation results in low accuracy of the output results of 3DCNN. The main reason is that the calculation of 3D convolution has a very high cost. This technical case successfully verifies the feasibility of 2D depth images as the prediction carrier of 3D geometric information, and the effectiveness of reinforcement learning in scene reconstruction tasks, bringing new ideas for image-based 3D scene reconstruction tasks.

1.4 Summary

This chapter first explains the basic principles and research status of 3D reconstruction methods from two directions: based on traditional multi-view geometry and based on deep learning and analyzes their advantages, disadvantages, and applicable scenarios. After the readers have a basic understanding of the background technology of 3D reconstruction, it shows specific technical cases from three directions: depth information estimation, 3D object reconstruction, and 3D scene reconstruction. In each case, basic task descriptions, knowledge point summaries, background technology summaries, technical case analyses, and experimental evaluations are carried out to help readers start from practice, strengthen their understanding of related theoretical knowledge, and be able to follow the related cases for hands-on practice.

However, there are many papers and engineering projects related to 3D reconstruction, and the author cannot introduce each aspect in detail. They can only start from specific cases that have been practiced, and attach relevant codes and links, and hope this can be helpful to readers. At the same time, we hope to attract more readers to join the research of 3D reconstruction and create more amazing results.

1.5 Discussion

1. Please enumerate some representation methods for 3D models other than 3D point clouds?
2. Please compare the advantages and disadvantages of Multi-view 3D Reconstruction and Deep Learning-based 3D Reconstruction.
3. Please briefly describe the basic process of visual SLAM and KinectFusion algorithms.
4. What are the benefits of using multi-task branches for training in depth prediction networks?
5. When using deep learning for 3D reconstruction, if there is no 3D model as the truth, how to train through video or multi-view pictures?

6. Practice: Build a depth prediction network for monocular images and train and evaluate it using the NYUDepth dataset.
7. Practice: Please attempt to construct a neural network that can input images from multiple views, output 3D voxels, and use the 3D models and rendered images in ShapeNet for training and evaluation.

References

1. NEWCOMBE R A, LOVEGROVE S J, DAVISON A J, et al. Dtam: dense tracking and mapping in real-time[C]//IEEE international conference on computer vision. New York: IEEE, 2011: 2320-2327.
2. STUHMER J, GUMHOLD S, CREMERS D. Real-time dense geometry from a handheld camera [C]//32nd annual symposium of the german association for pattern recognition. Berlin: Springer, 2010: 11-20.
3. GONZALEZ-JORGE H, RIVEIRO B, VAZQUEZ-FERNANDEZ E, et al. Metrological evaluation of Microsoft Kinect and Asus Xtion sensors [J]. Measurement, 2013, 46(6): 1800-1806.
4. LI W, WANG D, ZONG D, et al. RGBD-based robot indoor SLAM and path planning system [J]. Industrial Control Computer, 2016(3): 77-78.
5. SCHERER S A, ZELL A. Efficient ONBARD rgb-d slam for autonomous MAVs[C]//International conference on intelligent robots and systems. New York: IEEE, 2013: 1062-1068.
6. IZADI S, KIM D, HILLIGES O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera[C]//Proceedings of the 24th annual ACM symposium on user interface software and technology. New York: ACM, 2011: 559-568.
7. NEWCOMBE R A, IZADI S, HILLIGES O, et al. KinectFusion: real-time dense surface mapping and tracking [C]//10th IEEE/ACM international symposium on mixed and augmented reality. New York: IEEE, 2011: 127-136.
8. CHEN J, BAUTEMBACH D, IZADI S. Scalable real-time volumetric surface reconstruction [J]. ACM Transactions on Graphics, 2013, 32(4): 1-10.
9. NIESSNER M, ZOLLHOFER M, IZADI S, et al. Real-time 3D reconstruction at scale using voxel hashing [J]. ACM Transactions on Graphics, 2013, 32(6): 1-11.
10. HENRY P, KRAININ M, HERBST E, et al. Rgb-d mapping: using kinect-style depth cameras for dense 3D modeling of indoor environments [J]. International Journal of Robotics Research, 2012, 31(5): 647-663.
11. KELLER M, LEFLOCH D, LAMBERS M, et al. Real-time 3D reconstruction in dynamic scenes using point-based fusion[C]//International conference on 3D vision. New York: IEEE, 2013: 1-8.
12. NEWCOMBE R A, FOX D, SEITZ S M, et al. DynamicFusion: reconstruction and tracking of non-rigid scenes in real-time[C]//IEEE conference on computer vision and pattern recognition. New York: IEEE, 2015: 343-352.
13. DAI A, NIESSNER M, ZOLLHOFER M, et al. BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration[J]. ACM Transactions on Graphics, 2017, 36(3): 1-18.
14. ZHANG J, ZHU C, ZHENG L, et al. ROSEFusion: random optimization for online dense reconstruction under fast camera motion[J]. ACM Transactions on Graphics, 2021, 40(4): 17.
15. WU J, ZHANG C, XUE T, et al. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling[C]//30th conference on neural information processing systems. La Jolla: NIPS, 2016: 82-90.
16. GIRDHAR R, FOUHEY D F, RODRIGUEZ M, et al. Learning a predictable and generative vector representation for objects[C]//14th European conference on computer vision. AMSTERDAM: Springer International Publishing Ag, 2016: 484-499.

17. DAI A, QI C, NIESSNER M, et al. Shape completion using 3D-encoder-predictor CNNs and shape synthesis[C]//30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 6545-6554.
18. SONG S, YU F, ZENG A, et al. Semantic scene completion from a single depth image[C]//30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 190-198.
19. TATARCHENKO M, DOSOVITSKIY A, BROX T, et al. Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs[C]//16th IEEE international conference on computer vision. New York: IEEE, 2017: 2107-2115.
20. FAN H, SU H, GUIBAS L, et al. A point set generation network for 3D object reconstruction from a single image[C]//30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 2463-2471.
21. GROUEIX T, FISHER M, KIM V G, et al. A papier-mache approach to learning 3D surface generation[C]//31st IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2018: 216-224.
22. PARK J, FLORENCE P, STRAUB J, et al. DeepSDF: learning continuous signed distance functions for shape representation[C]//32nd IEEE/CVF conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2019: 165-174.
23. MESCHEDER L, OECHSLE M, NIEMEYER M, et al. Occupancy networks: learning 3D reconstruction in function space[C]//32nd IEEE/CVF conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2019: 4455-4465.
24. CHANG A X, FUNKHOUSER T, GUIBAS L, et al. ShapeNet: an information-rich 3D model repository [EB/OL] <https://arxiv.org/abs/1512.03012>.
25. XIAO J X, OWENS A, TORRALBA A, et al. SUN3D: a database of big spaces reconstructed using SFM and object labels[C]//IEEE international conference on computer vision. New York: IEEE, 2013: 1625-1632.
26. DAI A, CHANG A, SAVVA M, et al. ScanNet: richly-annotated 3D reconstructions of indoor scenes[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York: IEEE, 2017: 5828-5839.
27. CHOY C, XU D, GWAK J, et al. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction[C]//Proceedings of the European conference on computer vision. Berlin: Springer, 2016: 628-644.
28. KAR A, HÅNE C, MALIK J. Learning a multi-view stereo machine[C]//Proceedings of the annual conference on neural information processing systems, La Jolla: NIPS, 2017(30): 365-376.
29. YANG X, WANG Y, WANG Y, et al. Active object reconstruction using a guided view planner[C]//Proceedings of the international joint conference on artificial intelligence. Freiburg: IJCAI, 2018: 4965-4971.
30. KIM V G, LI W, MITRA N J, et al. Learning part-based templates from large collections of 3D shapes[J]. ACM Transactions on Graphics, 2013, 32(4): 1-12.
31. SHEN C, FU H, CHEN K, et al. Structure recovery by part assembly [J]. ACM Transactions on Graphics, 2012, 31(6): 1-11.
32. LI L, SUNG M, DUBROVINA A, et al. Supervised fitting of geometric primitives to 3D point clouds[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2019: 2652-2660.
33. SHARMA G, LIU D, MAJI S, et al. ParseNet: A parametric surface fitting network for 3D point clouds[C]// Proceedings of the European conference on computer vision. Berlin: Springer, 2020: 261-276.
34. SAXENA A, CHUNG S, NG A. Learning depth from single monocular images[C]// Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2005: 1161-1168.
35. SAXENA A, SUN M, NG A Y. Learning 3D scene structure from a single still image[C]// Proceedings of the international conference on computer vision. Piscataway, NJ: IEEE, 2007: 1-8.

36. LAINA I, RUPPRECHT C, BELAGIANNIS V, et al. Deeper depth prediction with fully convolutional residual networks[C]//Proceedings of the international conference on 3D vision. Piscataway, NJ: IEEE, 2016: 239-248.
37. CAO Y, WU Z, SHEN C. Estimating depth from monocular images as classification using deep fully convolutional residual networks [J].IEEE Transactions on Circuits and Systems for Video Technology, 2017, 28(11): 3174-3182.
38. GARG R, BG V K, CARNEIRO G, et al. Unsupervised CNN for single view depth estimation: geometry to the rescue[C]//Proceedings of the European conference on computer vision. Berlin: Springer, 2016: 740-756.
39. ZORAN D, ISOLA P, KRISHNAN D, et al. Learning ordinal relationships for mid-level vision[C]//Proceedings of the international conference on computer vision. Piscataway, NJ: IEEE, 2015: 388-396.
40. CASSER V, PIRK S, MAHJOURIAN R, et al. Depth prediction without the sensors: leveraging structure for unsupervised learning from monocular videos[C]//Proceedings of the AAAI conference on artificial intelligence. Palo Alto, CA: AAAI, 2019: 8001-8008.
41. EIGEN D, FERGUS R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture[C]//Proceedings of the international conference on computer vision. Piscataway, NJ: IEEE, 2015: 2650-2658.
42. MOUSAVIAN A, PIRSAVASH H, KOSECKA J. Joint semantic segmentation and depth estimation with deep convolutional networks[C]//Proceedings of the international conference on 3D vision. Piscataway, NJ: IEEE, 2016: 611-619.
43. WANG P, SHEN X, RUSSELL B, et al. Surge: surface regularized geometry estimation from a single image[C]//Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2016: 172-180.
44. GODARD C, MAC AODHA O, FIRMAN M, et al. Digging into self-supervised monocular depth estimation[C]//Proceedings of the international conference on computer vision. Piscataway, NJ: IEEE, 2019: 3828-3838.
45. ZHANG Z, CUI Z, XU C, et al. Joint task-recursive learning for semantic segmentation and depth estimation[C]//Proceedings of the European conference on computer vision. Berlin: Springer, 2018: 235-251.
46. ZHANG Z, CUI Z, XU C, et al. Pattern-affinitive propagation across depth, surface normal and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2019: 4106-4115.
47. SILBERMAN N, HOIEM D, KOHLI P, et al. Indoor segmentation and Support inference from RGBD images[C]//12th European Conference on Computer Vision(ECCV). BERLIN: Springer-Verlag Berlin, 2012: 746-760.
48. KINGMA D P, WELING M. Auto-encoding variational bayes[EB/OL]. (2013-12-20). <https://arxiv.org/abs/1312.6114>.
49. GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]// Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2014: 2672-2680.
50. FIORAIO N, KONOLIGE K. Realtime visual and point cloud slam[C]//Proceedings of the RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras. Berlin: Springer, 2011, 27: 1-8.
51. XU K, HUANG H, SHI Y, et al. Autoscanning for coupled scene reconstruction and proactive object analysis [J]. ACM Transactions on Graphics, 2015, 34(6): 1-14.
52. DENZLER J, BROWN C M. Information theoretic sensor data selection for active object recognition and state estimation [J]. IEEE transactions on pattern analysis and machine intelligence, 2002, 24(2): 145-157.
53. HUBER M F, DENCKER T, ROSCHANI M, et al. Bayesian active object recognition via Gaussian process regression[C]//IEEE International Conference on Information Fusion. Piscataway, NJ: IEEE, 2012: 1718-1725.
54. MNIIH V, HEES N, GRAVES A. Recurrent models of visual attention[C]//Advances in neural information processing systems. La Jolla: NIPS, 2014: 2204-2212.

55. XU K, SHI Y, ZHENG Y, et al. 3D attention-driven depth acquisition for object identification [J]. *ACM Transactions on Graphics*, 2016, 35(6): 1-14.
56. SU H, MAJI S, KALOGERAKIS E, et al. Multi-view convolutional neural networks for 3D shape recognition[C]//*Proceedings of the IEEE international conference on computer vision*. Piscataway, NJ: IEEE, 2015: 945-953.
57. NEALEN A, IGARASHI T, SORKINE O, et al. Laplacian mesh optimization[C]//*Proceedings of the 4th international conference on computer graphics and interactive techniques in Australasia and Southeast Asia*. New York: Association for Computing Machinery, 2006: 381-389.
58. KAZHDAN M, BOLITHO M, HOPPE H. Poisson surface reconstruction[C]//*Proceedings of the fourth eurographics symposium on geometry processing*. Goslar: Eurographics Association, 2006: 61-70.
59. THRUN S, WEGBREIT B. Shape from symmetry[C]//*IEEE international conference on computer vision*. Piscataway, NJ: IEEE, 2005: 1824-1831.
60. NAN L L, XIE K, SHARF A. A search-classify approach for cluttered indoor scene understanding [J]. *ACM Transactions on Graphics*, 2012, 31(6), 137: 1-137: 10.
61. KIM Y M, MITRA N J, YAN D M, et al. Acquiring 3D indoor environments with variability and repetition [J]. *ACM Transactions on Graphics*, 2012, 31(6), 138: 1-138: 11.
62. PAULY M, MITRA N J, GIESEN J, et al. Example-based 3D scan completion[C]// *Proceedings of the fourth eurographics symposium on geometry processing*. Goslar: Eurographics Association, 2005: 23-32.
63. CHEN K, LAI Y, WU Y, et al. Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information [J]. *ACM transactions on graphics*, 2014, 33(6): 1-12.
64. FOUHEY D F, GUPTA A, HEBERT M. Data-driven 3D primitives for Single Image Understanding [C] //*Proceedings of the IEEE international conference on computer vision*. Piscataway, NJ: IEEE, 2013: 3392-3399.
65. QI C, SU H, MO K, et al. PointNet: deep learning on point sets for 3D classification and segmentation [C] //*Proceedings of the IEEE conference on computer vision and pattern recognition*. Los Alamitos, CA: IEEE Computer Society, 2017: 77-85.
66. CHEN Y, MEDIONI G. Object modelling by registration of multiple range images [J]. *Image and vision computing*, 1992, 10(3): 145-155.
67. QI C, YI L, SU H, et al. PointNet++: deep hierarchical feature learning on point sets in a metric space [C]//*Advances in neural information processing systems*. La Jolla: NIPS, 2017: 5099-5108.
68. LI Y, BU R, SUN M, et al. PointCNN: convolution on x-transformed points [C]//*Advances in neural information processing systems*. La Jolla: NIPS, 2018: 828-838.
69. ZHANG L, WANG L, ZHANG X, et al. Semantic scene completion with dense CRF from a single depth image [J]. *Neurocomputing*, 2018, 318: 182-195.
70. SONG S, YU F, ZENG A, et al. Semantic scene completion from a single depth image [C]// *Proceedings of the IEEE conference on computer vision and pattern recognition*. Los Alamitos, CA: IEEE Computer Society, 2017: 1746-1754.
71. LIU S, HU Y, ZENG Y, et al. See and think: Disentangling semantic scene completion [C]// *Advances in neural information processing systems*. La Jolla: NIPS, 2018: 261-272.
72. ZHOU T, BROWN M, SNAVELY N, et al. Unsupervised learning of depth and ego-motion from video [C] //*Proceedings of the IEEE conference on computer vision and pattern recognition*. Los Alamitos, CA: IEEE Computer Society, 2017: 1851-1858.
73. GORDON A, LI H, JONSCHKOWSKI R, et al. Depth from videos in the wild: unsupervised monocular depth learning from unknown cameras [C]//*Proceedings of the IEEE/CVF international conference on computer vision*. Piscataway, NJ: IEEE, 2019: 8977-8986.
74. CASSER V, PIRK S, MAHJOURIAN R, et al. Depth prediction without the sensors: leveraging structure for unsupervised learning from monocular videos [C]//*Proceedings of the AAAI conference on artificial intelligence*. Palo Alto, CA: AAAI Press, 2019: 8001-8008.

75. DAI A, RITCHIE D, BOKELOH M, et al. ScanComplete: large-scale scene completion and semantic segmentation for 3D scans [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2018: 4578-4587.
76. WANG Y, TAN D, NAVAB N, et al. Forknet: multi-branch volumetric semantic completion from a single depth image [C]//Proceedings of the IEEE/CVF international conference on computer vision. Piscataway, NJ: IEEE, 2019: 8608-8617.
77. GARBADE M, CHEN Y T, SAWATZKY J, et al. Two stream 3D semantic scene completion [C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. Los Alamitos, CA: IEEE Computer Society, 2019: 416-425.
78. GUO Y, TONG X. View-volume network for semantic scene completion from a single depth image [C]//Proceedings of the international joint conference on artificial intelligence. Freiburg, Germany: IJCAI, 2018: 726-732.

Chapter 2

Scene Active Exploration



2.1 Overview

Autonomous exploration of unknown environments has been a topic of significant interest, as effective scene exploration can provide robust and generalizable versatile planning strategies for various downstream tasks, such as autonomous scene scanning, reconstruction, and visual semantic navigation. However, due to challenges like occlusion and the limited field of view of cameras, the robot's observation of the environment usually only involves a local area of the entire environment (e.g., a corner of a room). This limitation makes it difficult for the robot to plan its actions effectively based solely on such partial observations. Therefore, in addition to the robot's local observations, scene exploration also necessitates rich contextual knowledge about indoor environments. Specifically, the robot can utilize functional priors of the indoor environment to guide high-level search strategies (e.g., beds are usually found in bedrooms) and spatial layout priors to assist with low-level path planning (e.g., planning the shortest collision-free path to reach a target point). Representation methods for these scene priors include building 3D object model libraries for model matching, constructing scene knowledge graphs and using Graph Convolutional Networks (GCNs) for feature extraction, and employing scene completion algorithms to explicitly model prior scene information. The core challenge is how to effectively integrate the scene's local observation data with contextual prior knowledge to progressively improve the robot's understanding of the global scene, thus guiding more informed path planning and completing tasks like autonomous scene reconstruction and navigation. Consequently, the main challenges in scene exploration involve how to construct an effective representation method for scene prior knowledge, how to intelligently analyze local observation data in combination with contextual information, and how to find a balance between global scene exploration and local scanning, all with the goal of incrementally enhancing the robot's comprehension of the global scene space. This chapter reviews the current research status both domestically and internationally, outline the general

approaches to scene exploration, and provide detailed case studies of two representative works as technical examples.

For related literature and code files of this chapter, the following SN-branded shortURLs can be scanned by the SN More Media app.

(▶ sn.pub/4mojrl)

2.2 Related Work

2.2.1 Scene Exploration in 3D Reconstruction

Autonomous scanning and reconstruction of unknown scenes have long been critical research topics. To achieve autonomous 3D scanning and reconstruction of unknown indoor environments, it is essential to strike a balance between rapid scanning for global scene exploration and high-quality scanning for detailed reconstruction. Traditional global scene exploration methods primarily rely on spatial information, making the acquisition of comprehensive scene data through autonomous robot navigation a key aspect. On the other hand, high-quality scanning for scene reconstruction depends on capturing the local geometric details of visible surfaces, requiring precise robot movements to ensure the scan's completeness and stability and to obtain continuous frames that reduce scanning and reconstruction errors. One effective approach is to integrate scene exploration and reconstruction, thereby enabling autonomous scene reconstruction. The main challenge in this approach lies in balancing the needs for fast global exploration and meticulous, high-quality reconstruction.

2.2.1.1 Autonomous 3D Scene Reconstruction Guided by Time-Varying Tensor Fields

In this work, Xu et al. [1] proposed a novel method to address the challenge of autonomous 3D scene reconstruction by integrating navigation with reconstruction. The proposed approach involves planning the robot's motion path to simultaneously consider the global efficiency required for rapid exploration and the local smoothness necessary for capturing high-quality scans. The operation of the RGB-D camera, mounted on the mobile robot's mechanical arm, is determined by the required reconstruction quality as well as the robot's own movement. The core concept of this method is to guide the robot's motion using a time-varying tensor field, thereby solving the problem of 3D camera control under the constraints of the robot's 2D motion path. The tensor field is updated in real-time, aligning well with the progressive reconstruction process. Experiments have demonstrated that tensor fields are highly effective for guiding autonomous exploration and scanning-based reconstruction for two main reasons: First, they feature sparse and controllable singular

points, which enable the generation of locally smooth robot paths. Second, their topological structure facilitates globally efficient path routing in local reconstruction scenes. As a result, this method effectively balances exploration efficiency with reconstruction quality, addressing the trade-offs between global exploration and local scanning. Tests with mobile robots have further validated that this approach enables smooth exploration and high-quality reconstruction of unknown indoor scenes.

2.2.1.2 Object-Aware Guidance for Autonomous Scene Reconstruction

In this work, Liu et al. [2] proposed a novel method that enhances object-aware guidance for the exploration, reconstruction, and understanding of an unknown scene within one navigation pass. The method alternates between object analysis, which identifies the next best object (NBO) for global exploration, and object-aware information gain analysis, which determines the next best view (NBV) for local scanning. First, an object-based segmentation method is introduced, minimizing multi-class graph cuts to extract semantic objects from the current scene surface. Next, the object of interest (OOI) is identified as the NBO that the robot needs to visit and scan. Finally, the OOI is precisely scanned from the viewpoint determined by the NBV strategy. When the OOI is fully identified as a complete object, it is replaced with the most similar 3D model from a shape database. This algorithm iterates until all objects in the scene are identified and reconstructed. Liu et al.'s method adopts an object-aware guidance for scene exploration, progressively achieving global exploration by scanning the objects in the scene one by one. For each object, the NBV strategy is employed to perform detailed reconstruction, closely mimicking how humans scan and reconstruct a scene. This approach enables effective and comprehensive scene exploration.

2.2.1.3 Active Scene Understanding Guided Semantic 3D Reconstruction

In this work, Zheng et al. [3] proposed a novel approach for enabling robots to actively understand unknown indoor scenes through online RGB-D reconstruction and semantic segmentation. This method drives the exploratory robot's scanning process by focusing on the recognition and segmentation of semantic objects within the scene, aiming to enhance the identification and categorization of these objects. The algorithm is built on a voxel depth fusion framework, similar to the KinectFusion algorithm, which allows for real-time semantic labeling of the voxels during the online reconstruction process. The robot is guided by an online estimated discrete viewing score field (VSF), which is parameterized on the 3D space of 2D location and azimuth rotation. The VSF stores the score for each corresponding viewpoint within a grid, which is used to measure how effectively it reduces the uncertainty (entropy) in both geometric reconstruction and semantic labeling. The next best view (NBV) is then selected based on the VSF as the target for each time step. By

maximizing the integral viewing score (information gain) along the path and camera trajectory, the walking path and camera trajectory between two adjacent NBVs are jointly optimized. Zheng et al.’s approach guides the robot in selecting the scanning viewpoint by focusing on reducing uncertainty in the scene. Specifically, the robot is directed to scan from viewpoints that maximize the reduction of both reconstruction and semantic uncertainties. Once there are no more uncertain areas in the reconstructed scene, it indicates that the robot has completed the scene exploration and reconstruction process.

2.2.2 Scene Exploration in Visual Semantic Navigation

The visual semantic navigation task involves generating a series of actions that guide a robot from a random starting point in an unknown environment to a specified target object category (such as a bathroom). To accomplish this task, the algorithm must be capable of both locating and navigating to an instance of the designated object category. Unlike traditional point-goal navigation, where the coordinates of the target are known and the robot’s sole task is to find a collision-free path to those coordinates, visual semantic navigation requires the robot to simultaneously answer two questions: First, where should the robot go (i.e., identifying the location of the target object)? And second, how should it get there (i.e., planning an effective route to reach the target object)?

Due to occlusions and the limited field of view of the camera, a robot’s observation of an environment often only captures a small portion of the entire space (e.g., a corner of a room in a large house). Therefore, in addition to the robot’s local observation data, visual semantic navigation also requires rich contextual prior knowledge about indoor environments. Specifically, the robot can use functional priors of indoor environments to guide high-level search strategies (e.g., beds are typically found in bedrooms) and utilize spatial layout priors to assist in low-level behavior planning (e.g., determining how to leave a room without collisions). These contextual priors help the robot explore the scene, enabling it to infer whether the target object might be present in unobserved areas (such as occluded or invisible regions) and assess the likelihood of the target object being in those areas. This helps the robot more effectively locate the spatial position of the target object, addressing the question of where to go. Additionally, by reasoning about the possible spatial layout in unobserved areas using contextual priors, the robot can plan a collision-free path to the target location, solving the problem of how to reach the target.

2.2.2.1 Semantic Scene Completion Guided Visual Semantic Navigation

In the work presented by Liang et al. [4], the focus is on visual semantic navigation, where a robot autonomously generates actions to navigate to a specified target object category within an unknown environment. To accomplish this task, the

algorithm needs to both locate and navigate to an instance of the specified object category. Unlike traditional point-goal navigation, this task requires the robot to have a stronger understanding of contextual priors in indoor environments. Based on this requirement, Liang et al. employed a confidence-aware semantic scene completion module to explicitly model scene priors, allowing the robot to complete the scene and guide its navigation planning. Given the robot's local observation of the environment, the algorithm first infers a complete scene representation that includes semantic labels for unobserved areas and a confidence map that evaluates the reliability of these predictions. The navigation strategy network then infers the next actions from the completed scene and confidence maps to achieve the visual semantic navigation task. By learning statistical data on the layouts of multiple rooms and using contextual clues, the robot can predict the complete room layout beyond its current field of view, even in an unfamiliar scene. This prediction helps the robot understand its environment, guiding it to explore the scene effectively, determine the location of the target object, and plan subsequent actions based on scene priors to complete the navigation task.

2.2.2.2 Scene Prior Guided Visual Semantic Navigation

In literature [5], inspired by how humans navigate toward target objects in new environments, Wei et al. explored the use of semantic priors to accomplish both semantic and goal-oriented navigation tasks. Specifically, when humans search for target objects in unknown settings, they rely on semantic and functional priors developed over time to effectively guide their search and navigation efforts. For instance, people looking for a cup would typically search in cabinets near a coffee machine, while those looking for fruit might check inside a refrigerator. These observations highlight that prior knowledge about the world's semantic and functional structures enhances navigation efficiency. Building on this insight, the study focused on integrating semantic priors into semantic navigation tasks using graph convolutional networks (GCNs). These networks incorporate prior knowledge into a deep reinforcement learning framework, enabling robots to use features from a knowledge graph to predict actions. Consequently, Wei and his team designed and implemented an actor-critic model that leverages knowledge graphs and object visibility information. This model uses a GCN network to embed prior knowledge into the deep reinforcement learning framework. The robot's knowledge is encoded in a graph, with the GCN allowing for efficient encoding of any structured graph. The robot's knowledge is updated based on its observations of the current environment, as well as the knowledge or prior knowledge from previous steps, with the prior knowledge being derived from a large-scale dataset designed for scene understanding. During navigation, the robot utilizes its current observations, the given target, and its prior knowledge to reason about the environment, much like a human would. This reasoning allows the robot to explore environmental clues related to the target object, guiding its search throughout the scene to successfully locate the target, thereby completing the navigation task.

2.2.2.3 Topological Graph Structure Guided Visual Navigation

In literature [6], the image-goal navigation problem was investigated, which entails navigating to the location indicated by a target image within a new, unseen environment. To address this issue, Singh and colleagues developed a spatial topological representation that effectively utilizes semantic information and approximates geometric reasoning, avoiding the traditional metric map representation, which is more susceptible to positioning errors and noise. The core of this approach is the use of nodes with associated semantic features. Each node is represented intuitively by a 360° panoramic image, providing local geometric context. The nodes are interconnected using approximate relative poses. Furthermore, two-directional functions, F_g and F_s , were introduced to extract the geometric and semantic attributes of the nodes. Specifically, F_g estimates the likelihood that the robot will encounter free space, while F_s estimates the likelihood of encountering the target image if the robot moves in a certain direction. By explicitly modeling and learning the F_s function, the method ensures that structural priors are encoded and utilized during exploration and navigation in entirely new, unseen environments. Compared to traditional methods and end-to-end learning approaches, this method offers several advantages: (1) uses a graph-based representation, allowing for effective long-term planning; (2) explicitly encodes structural priors through the F_s function; (3) the geometric function F_g allows for effective exploration and online map building for new environments; (4) all functions and strategies can be learned in a fully supervised manner, without the need for unreliable credit allocation through reinforcement learning (RL). The algorithm is robust under noise-driven conditions, as it can be built, maintained, and used effectively. Experimental results demonstrate that this method constructs an efficient representation, capable of capturing structural regularities and effectively solving long-term navigation challenges.

2.2.3 Summary

This chapter primarily introduces various scene exploration strategies within the context of autonomous scene reconstruction tasks and visual semantic navigation tasks, while also briefly outlining the exploration strategies in several related studies. In the field of autonomous scene reconstruction, the challenge lies in balancing global exploration with local scanning to achieve an optimal scanning path and high-quality reconstruction results. One approach discussed involves object-aware guidance for scene exploration, where the robot sequentially scans objects within the scene, allowing these objects to guide the global exploration process. Another method highlighted is based on the uncertainty of scene reconstruction, where the scene is explored according to areas of uncertainty within the reconstruction model. When the model no longer contains uncertain factors, the scanning process is considered complete. In contrast to reconstruction tasks, scene exploration in visual semantic navigation focuses less on globally traversing the entire scene. Instead, it

emphasizes using contextual information and prior knowledge of the scene layout to infer and predict the structure of unseen areas. This approach helps in “guessing” the location of the target object and the layout of the unknown environment, while also planning a reasonable path to reach the target. Chapter 3 further elaborates on scene exploration techniques by introducing three technical cases that detail the specific implementation processes in 3D reconstruction and visual semantic navigation tasks, along with the presentation of experimental results.

2.3 Technology Practice

2.3.1 Case Study: Object-Aware Guidance for Autonomous Scene Reconstruction

2.3.1.1 Task Description

To carry out autonomous 3D scanning and online reconstruction of unknown indoor scenes, one has to find a balance between global exploration of the entire scene and local scanning of the objects within it. In this work, we propose a novel approach, which provides object-aware guidance for autoscanning, for exploring, reconstructing, and understanding an unknown scene within one navigation pass. Our approach interleaves between object analysis to identify the next best object (NBO) for global exploration, and object-aware information gain analysis to plan the next best view (NBV) for local scanning.

In this technical case, a new scene exploration method is proposed, namely, object-aware guidance for autonomous scene reconstruction. This method allows the robot to complete global path planning and local scanning viewpoint planning simultaneously in one scanning process. The robot identifies semantic objects, visits them one by one, and actively scans the currently visited objects as shown in Fig. 2.1. In each column, the object of interest (OOI) is marked with a rectangular

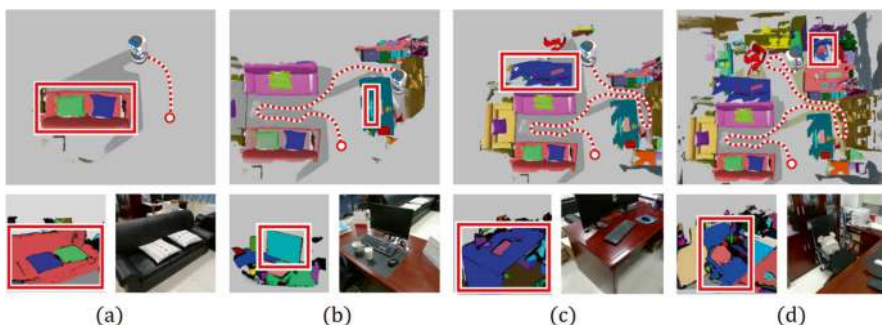


Fig. 2.1 Autonomous scene scanning reconstruction of a real office scene using the object-aware guidance approach

frame. The top row shows the navigation path (dashed line), previous scanning viewpoints (shown as dots), and the robot's current position, with different colored objects being the objects reconstructed in the scene; the bottom row shows the depth image (left) and RGB image (right) from the robot's current viewpoint. This method performs global path planning and local viewpoint planning in real time during a single navigation process and obtains a reconstructed scene with semantic objects (d). This active scanning method is mainly inspired by the following observation: When people use handheld scanners to scan indoor scenes, they tend to identify an object and then complete its scanning before scanning the next object. Similarly, when the method proposed in this technical case explores the scene, the robot first determines the object with the highest recognizability from its current observation viewpoint, called the next best object (NBO), and sets it as the object of interest (OOI). Then, the robot visits and scans the OOI driven by its recognizability-increasing NBVs. After the robot completes the reconstruction of the current OOI, it continues to identify and scan the next OOI. This sequential visit and scan of all objects constitutes a navigation path for the robot in the scene.

The core of this method is to iteratively execute the “Recognition-Planning” algorithm, which alternates between object-level shape analysis to identify NBOs for global navigation and object-aware information gain analysis to plan NBVs for local scanning. First, this method proposes a model-driven object objectivity concept, which is based on the prior knowledge of 3D shapes provided by the 3D model database. Second, this method proposes an object-based segmentation method, which extracts objects through multi-class graph cut optimization, then the robot visits the identified OOIs and scans them from the viewpoint determined by the NBV strategy, which is based on increasing information gain to identify OOIs. When an OOI is identified as a complete object (with a very high objectivity score), the most similar 3D model is retrieved from the database and inserted into the scene to replace it. The algorithm alternates between NBO and NBV estimation until all objects in the scene are reconstructed, as shown in Fig. 2.2. This method

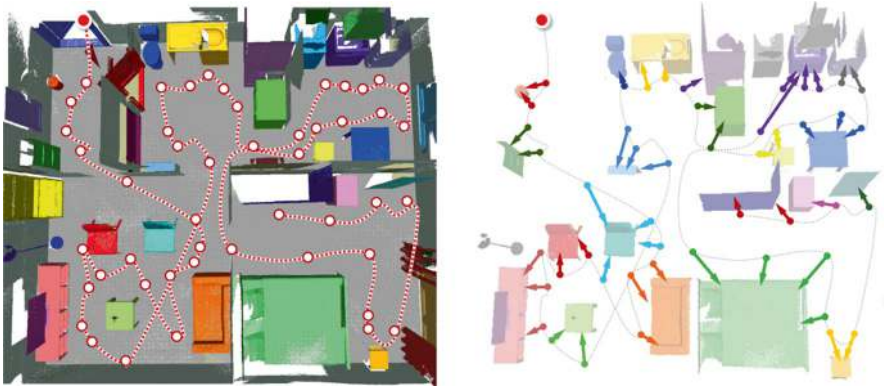


Fig. 2.2 Autonomous scanning reconstruction process of unknown scenes based on object-based segmentation method

demonstrates the global path (left) and local viewpoint (right) planning performed during autonomous scanning reconstruction of unknown scenes. The solid point in the upper left corner is the starting point where the robot enters the scene. The red dashed line in the left figure represents the navigation path of the robot in the scene. The hollow points on the path are the locations where the robot performs local scanning. Each arrow in the right figure shows the scanning viewpoint direction of the robot at its location, pointing to the corresponding object it is scanning. Arrows for the same object are shown in the same color. This method dynamically handles the scanning depth frame stream, so as to autonomously explore and semantically reconstruct unknown scenes in a one-time navigation process. Unlike the classic treasure-hunting problem in robotics, the goal of this method is to simultaneously scan the scene and identify various objects in it through online reconstruction and analysis.



Chapter Knowledge Points

Convex Set: In convex geometry, a convex set is a subset of an affine space that is closed under convex combinations. More specifically, in Euclidean space, a convex set is one for which every point on the line segment connecting any pair of points in the set is also in the set. For example, a cube is a convex set, but anything hollow or with indentations, like a crescent, is not a convex set.

Graph Cut: Graph cut is a very useful and popular energy optimization algorithm, widely used in computer vision for foreground-background segmentation (image segmentation), stereo vision, image matting, etc., and is currently widely used in the field of medical imaging. These methods associate the image segmentation problem with the minimum cut (min cut) problem of the graph.

Frontier: The map representation used by frontier-based algorithms is evidence grids. Evidence grids obtain the occupancy status of the environment through sensor distance data, provide detailed environmental feature data, and are an important basis for robot navigation and path planning. There are three types of cells in the evidence grid: free area, indicating that there are no obstacles in the cell; occupied area, indicating that there are obstacles in the cell; unknown area, indicating that the cell has not yet been perceived by the robot and belongs to an unknown area. The frontier is the boundary area between the free area and the unknown area. By continuously moving the robot to new frontiers, a complete scene map can be constructed even when no frontiers can be found.

2.3.1.2 Background Technology

- (1) **Autonomous Scene Exploration and Scanning:** With the emergence and rapid development of personal robots and commodity-grade depth cameras, a large amount of work has emerged that explores large-scale indoor scenes and scans individual objects through autonomous scanning systems [1, 7–9]. Autonomous scanning systems are usually equipped with a mobile robot with a fixed camera or a robotic arm with a depth camera [7–8, 10]. For large indoor scenes, the robot builds a scene map and locates its position in the map based on simultaneous localization and mapping (SLAM) technology while exploring the scene [11–13]. For individual objects, the robot should scan the objects with fine-grained geometric details, and then reconstruct their geometric structure with carefully planned scanning angles [9, 10, 14]. However, the systems for automatic scene scanning and automatic object scanning are usually separate, and there is almost no work that couples the scanning of scenes and objects in a unified navigation system. Reference [1] implemented autonomous scanning in one-time navigation, but their work lacks object recognition. This technical case proposes a novel autonomous scanning method that can complete the exploration of unknown scenes and the semantic reconstruction of objects in the scene by processing dynamically acquired depth data.
- (2) **Scene analysis and understanding:** Scene reconstruction requires high-level analysis and understanding of the objects in the scene and their spatial relationships to characterize the object composition and spatial structure of the scene [12, 15–16]. In recent years, data-driven methods have attracted more attention because they use 3D model databases with object-level segmentation and semantic labels to assist in object extraction, understanding, and analysis, and can extract the structure and contextual relationships between objects [12, 14, 17–18]. This technical case also adopts a data-driven method, specifically, retrieving similar models from the database as shape priors for recognizing and identifying objects in the scene and using them as guidance for the robot's global path planning and local scanning angle planning. Unlike methods that rely on robot interaction for object extraction [19], the method used in this technical case is achieved through object recognition.
- (3) **Global Path Planning:** In order to obtain the global structure of the scene, the robot's path planning must be integrated with robust camera positioning and SLAM scanning registration to run together, which can be achieved by jointly minimizing the uncertainty of scene mapping and camera positioning [20]. There are already many methods for this problem, most of which require ensuring sufficient data overlap for frame-to-frame registration. Some methods focus on dealing with drift or loop closure problems caused by registration errors [21–25]. Xu et al. [1] proposed a method based on time-varying tensor fields that can simultaneously compute the smooth motion of the robot path and camera trajectory. However, this method can only perform pass-through scanning with the movement of the robot and cannot perform detailed scanning of individual objects, and it does not perform object recognition. Different from

existing works, this approach identifies the objects and uses them as a high-level guidance to explore the scene, which is inspired from the object-aware attention mechanism that humans focus attention selectively on identified objects being observed

- (4) **Local Viewpoint Planning:** The choice of the direction of the robot sensor's viewpoint is crucial for autonomous scanning to obtain the geometric shape of 3D objects. Its goal is to reduce the uncertainty of recognition and reconstruct the surface of the object with the fewest scanning angles. There are already many existing next best view (NBV) algorithms to actively acquire and scan 3D objects [9, 14] as well as 3D scenes [19, 26–27]. Unlike existing methods, this technical case uses 3D models retrieved from the database to shape prior information and performs object-aware information gain analysis to plan the NBV for local scanning. The key to this method is to implement object recognition and scanning angle planning in a coupled way.

2.3.1.3 Technical Implementation

Figure 2.3 is a pipeline of the object-guided autonomous scene scanning and reconstruction method. The movement of the robot Ω is described as a series of steps at its location or its sensor viewpoint. After robot Ω enters an unknown indoor room, it begins to acquire the original RGB-D data stream as input, and the underlying RGB-D SLAM framework fuses the acquired depth data with the current scene surface T . S (initially $S = \emptyset$) is represented as the reconstructed scene, which includes previously scanned objects from previous steps, and the 3D indoor model database is M .

At each step, the current scene surface T is first segmented into a set of pre-segmented near-convex components (Fig. 2.3a), objectness computation (Fig. 2.3b), and the pre-segmented components are merged into a set of post-segmented objects by minimizing multi-class graph cuts (Fig. 2.3c). These post-segmented objects serve as high-level guidance for planning the movement of robot Ω . Robot Ω selects the most prominent object as OOI γ (Fig. 2.3d), then moves to OOI γ and begins to actively scan it. With the assistance of models in the 3D indoor model database M that are similar to OOI γ , the NBV V is selected for the next observation and scan

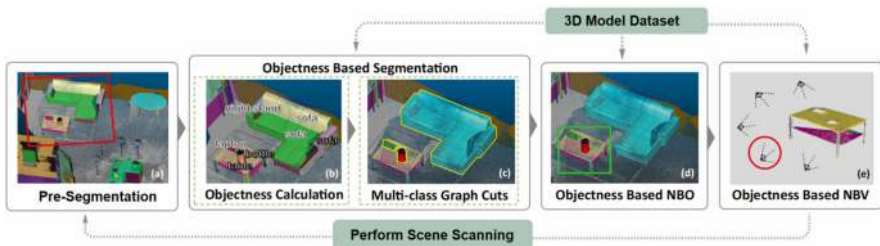


Fig. 2.3 Pipeline of object-guided autonomous scene scanning and reconstruction method

(Fig. 2.3e). When OOI γ is recognized as a complete object (with a very high objectness score), it is replaced by the most similar 3D model in the 3D indoor model database M .

This process is repeated until all post-segmented objects in the current scene surface T have been processed, resulting in a complete scene reconstruction result S . The sample code summarizes the entire automatic scanning process.



Sample Code

Input: Initial viewpoint of robot V

Output: Reconstructed scene S

1. Initialization: The initial reconstructed scene is empty, and the depth data under the initial viewpoint V is reconstructed to obtain T ;
2. REPEAT;
3. Pre-segment T to get C ;
4. Post-segment C to get R ;
5. Select NBO from R to get NBO γ ;
6. Select the next best viewpoint NBV V for NBO γ ;
7. Fuse the reconstructed data obtained under the new viewpoint V with T ;
8. IF γ is a complete object THEN;
9. Directly fuse γ with S and remove the object γ from T ;
10. UNTIL T is empty.

Objectness-Based Segmentation

- (1) Pre-segmentation. (1) Underlying SLAM framework. Underlying our system, we run a GPU version of the dense RGB-D SLAM framework to record the robot's trajectories and camera transformations. (2) Pre-segmentation: An incremental segmentation algorithm [28] is run on top of the SLAM framework. It separately segments the acquired depth data, and then, through the estimated camera pose, gradually merges the obtained segments in a unified global segmentation map. This method can segment the scene surface T into near-convex components (such as the legs and armrests of a chair), called pre-segmented components, represented as $C = \{ci, i = 1, 2, \dots, nc\}$ (Fig. 2.3a). Please note that if some pre-segmented components are only partially scanned or the acquired data is incomplete due to occlusion, they may not exist as complete semantic components in the model.
- (2) Model-driven objectness. (1) 3D model database. This method constructs a 3D model database M (Fig. 2.4) to provide the robot Ω with prior knowledge of 3D shapes and endows it with the ability to recognize and identify semantic objects in the scene surface T . First, 3D indoor models M^* of nl categories, such as chairs, tables, sofas, beds, and bookshelves, are collected. Each category is assigned a label $L \in \{1, 2, \dots, nl\}$. Due to the low quality and noise of the 3D



Fig. 2.4 The database M of 3D indoor models

data captured by the depth sensor, this method converts these clean models into point data to support more accurate online local matching and object recognition. Specifically, each model $Z \in M^*$ is virtually scanned into 3D point data m , each with a label, represented as $L(Z)$. Secondly, the algorithm is used to segment the 3D point data m into pre-segmented components [28]. In addition, for any pair of adjacent pre-segmented components of the 3D point data m , they are merged into a larger component. Finally, the 3D point data m , pre-segmented components, and adjacent component pairs with the same label $L(Z)$ are all put

into the 3D model database M . Note that this method puts all components and adjacent component pairs into M because the merging of two adjacent components allows graph cut optimization to merge multiple adjacent pre-segmented components in the pre-segmented component set C into a more complete object, thereby significantly enhancing object confirmation and recognition.

(2) Similar model set. For each $c \in C$, we search for a few models in M that are most similar to c and use them as candidates for recognition. As c might be incomplete due to occlusion, we adopt a partial matching method to find similar models. First, we uniformly sample np key points ($np = 500$) from each $m \in M$ and c , respectively, using the Lloyd's algorithm [29]. Second, we utilize a learning-based 3D shape descriptor, 3DMatch [25], which is learnt from real scanned depth data of indoor scenes. Then, we assign each key point with the three closest cluster centers according to the codebook of M and compute histograms of every cluster center. In addition, we use a spatially sensitive bag of words (BoW) to overcome the drawback by considering the spatial position among the key points. Thus, we can quickly compute a similar model set $M(c) = \{m_1, m_2, \dots, m_{ns}\} \subset M$ with ns models from M that are most similar to c ($n_s = 5$ in our implementation)

(3) Objectness: This method proposes a model-driven objectness concept, which is based on the prior knowledge of 3D shapes provided by the 3D model database, and calculates the objectness score of the scanned object according to this 3D shape prior, which is used as the basis for selecting NBO and determining whether the object has been scanned (for the specific calculation of objectness, please refer to reference [2], which is not detailed here).

- (3) Post-segmentation (based on objectness segmentation). Based on objectness, a post-segmentation technique is further proposed to refine pre-segmentation and obtain object-level segmentation. This is a segmentation based on objectness, which is achieved by combining objectness measurement with recognition rate in multi-class graph cut optimization, thereby effectively coupling segmentation and recognition in the same optimization method. This also allows the post-segmentation method to effectively merge adjacent pre-segmented components into a more complete object, as shown in Fig. 2.5. The goal of post-segmentation is to assign a label $lc \in \{1, 2, \dots, nl\}$ to each pre-segmented component $c \in C$, so that adjacent components from the same object have the same label, allowing them to merge into a more complete object (for the specific post-segmentation formula calculation, please refer to reference [2], which is not detailed here).

Object-Based Autonomous Scanning Reconstruction Strategy

Using object-based segmentation techniques, a set of post-segmented objects is obtained on the current scene surface T , represented as $R = \{r_1, r_2, \dots, r_{nr}\}$.

- (1) Next Best Object (NBO). The robot Ω needs to determine the OOI γ in the post-segmented object set R as the next best object for the robot Ω to visit. When the

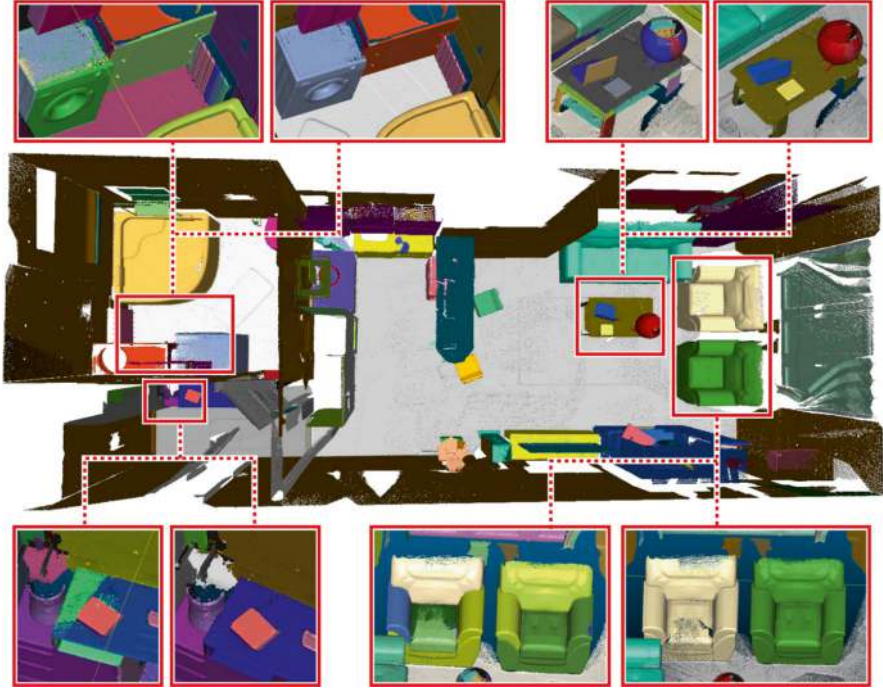


Fig. 2.5 Four regions in the scanned scene (middle row) are zoomed and their segmentation results of pre-segmentation (left) and post-segmentation (right) are shown side by side for comparisons

objectness and visual saliency scores are added together, the post-segmented object with the highest total (marked with a red rectangle) is selected as γ in R (Fig. 2.6).

$$\gamma = \arg \max_{r \in R} O(r) + S(r) \quad (2.1)$$

where objectness $O(r) = \max O(r, m)$, $m \in M(r)$ measures the degree to which the post-segmented object r in the 3D indoor model database M could be an object; visual saliency $S(r)$ is the saliency score of the post-segmented object r observed from the current observation direction V of the robot. The saliency score consists of three parts:

$$S(r) = wzSz(r) + weSe(r) + wdSd(r) \quad (2.2)$$

where the distance score $S_z(r) = \exp \left[- (C(r) - P_\Omega) / \max_{\bar{r} \in R} \left(C(\bar{r}) - P_\Omega \right) \right]$ measures

the distance from the center $C(r)$ of the post-segmented object r to the robot's position P_Ω . The direction score $Se(r) = [C(r) - P_\Omega]V$ measures the angle between the direction of the pre-segmented component c and the viewing direction of the robot Ω . The size score $S_d(r) = A(r) / \max_{\bar{r} \in R} A(\bar{r})$ measures the size of the post-segmented

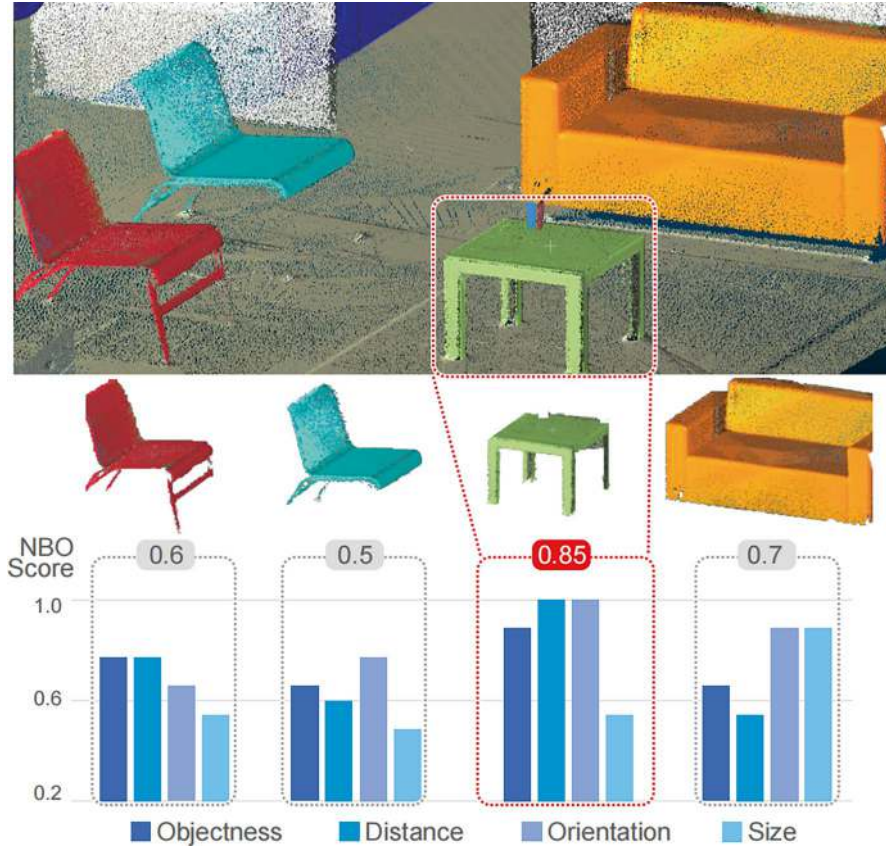


Fig. 2.6 The NBO scores of four post-segmented objects (see color figure in front of the text)

object r , where $A(r)$ is the area of the post-segmented object r , and w_z , w_e , w_d are weights (default $w_z=1.5$, $w_e=1$, $w_d=1$).

(2) Next Best View (NBV). Then the robot Ω moves to OOI γ and begins to actively scan it using the planned NBVs.

The candidate models in the similar model set $M(\gamma) = \{m_1, m_2, \dots, m_{ns}\}$ are used as shape priors to guide the robot to perform a comprehensive and complete scan of the object to which γ belongs. Therefore, by selecting the optimal view provided by $M(\gamma)$, the maximum conditional information gain is achieved, which not only reduces the recognition uncertainty of γ but also improves the recognition rate of all possible candidates.

(1) Candidate views: n_v points are uniformly sampled around the center of γ , represented as $\{V_1, V_2, \dots, V_{n_v}\}$, as candidate viewpoints for NBVs (this method sets $n_v = 16$). The goal is to select the next best viewpoint with the maximum conditional information gain as the NBV, which can provide better scanning to

improve the completeness and recognition rate of γ . Please note that some viewpoints may be invalid for γ because they are obscured or located inside another object (such as a wall). In this case, this method directly deletes these viewpoints.

- (2) TDF representation of object shape: The information gain of the voxels around γ is calculated with the help of the candidate views in the similar model set $M(\gamma)$. Therefore, all shapes are first converted into voxel representations, specifically, for shape $X \in \{\gamma\} \cup M(\gamma)$, the truncated distance function (TDF) on the voxel is defined as $f(x, X) = 1, x \in X$

$$f(x, X) = \begin{cases} 1, & x \in X \\ 0, & \text{others} \end{cases} \quad (2.3)$$

Here, x represents the voxel in the domain. To reduce the impact of data noise, this method uses Gaussian blur to fuse the distance field with its adjacent voxels. In addition, this method defines a visibility function of the voxel observed from viewpoint V_j . If voxel x can be directly seen from V_j without any obstruction, then $g(x, V_j) = 1$; otherwise, $g(x, V_j) = 0$.

- (3) Conditional information gain: For each candidate viewpoint $V_j (j = 1, \dots, m_v)$, this method combines conditional entropy and information gain with the help of all candidate models $m_i (i = 1, 2, \dots, n_s)$, as the conditional information gain of γ , as shown in Fig. 2.7. For the current OOI γ (top left), 3 candidate models Chair 1, Chair 2, and Table 1 (shown in different colors), represented as m_1, m_2, m_3 (top right), are retrieved from the database. These three candidate models serve as shape priors to guide the selection of NBV. After they are aligned with γ , some viewpoints are sampled (middle right), and the conditional information gain of γ is calculated for each viewpoint. For the viewpoint marked with a red circle, the information gain of all visible voxels (middle left) is calculated. The prior entropy $H(x)$ and conditional entropy $H(x|m_i)$ of the three candidate models are shown in the histogram (bottom) (here, three voxels are used as examples). The solution for conditional information gain is as follows:

$$\max_{j=1, \dots, n_s} G^j = \sum_{i=1}^{n_s} p(m_i) G^j(m_i) \quad (2.4)$$

where $p(m_i) = O(\gamma, m_i) / \sum_{k=1}^{n_s} O(\gamma, m_k)$ is the probability of m_i being the best candidate for γ , $G^j(m_i)$ is the information gain of all visible voxels in m_i from viewpoint V_j . $G^j(m_i)$ is defined as

$$\sum_{x \in \Delta} [H(x) - H(x|m_i)] \quad (2.5)$$

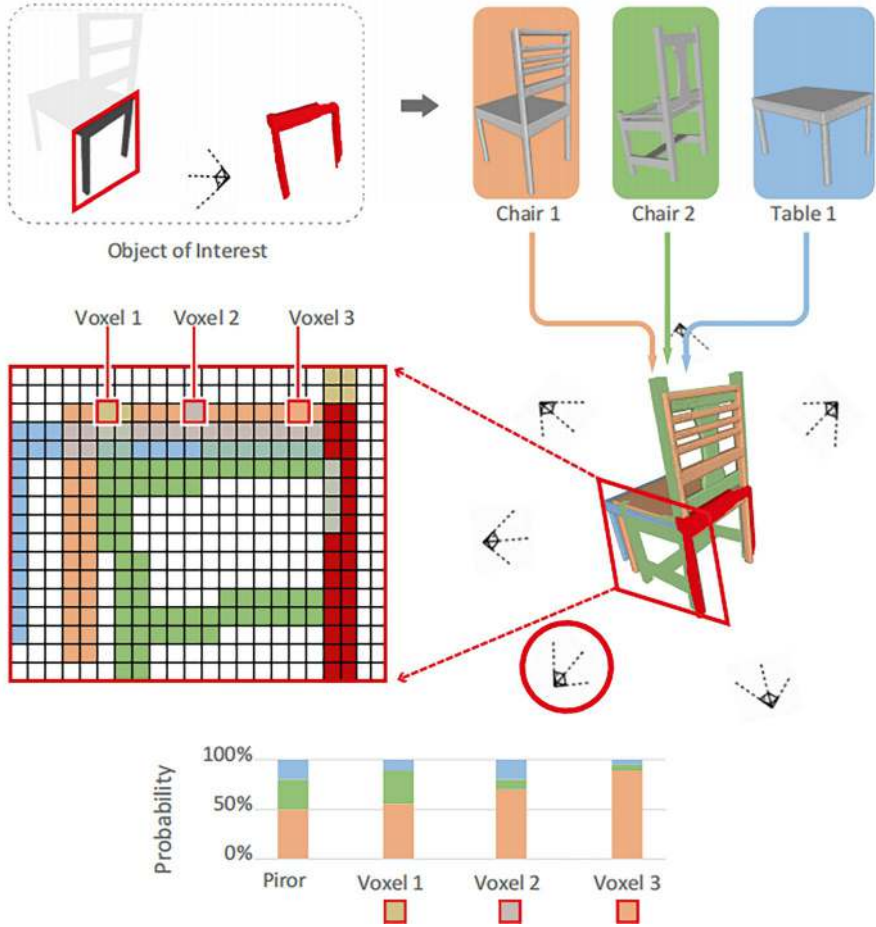


Fig. 2.7 An illustration of the computation of NBV (see color figure before the text)

where Δ is the set of voxels on m_i , not on γ , and visible from viewpoint V_j , i.e.,

$$\Delta = \{x \mid f(x, m_i) \neq 0, f(x, \gamma) = 0, g(x, V_j) = 1\} \quad (2.6)$$

$$H(x) = -\sum_{i=1}^{n_x} p_x(m_k) \log p_x(m_k) \quad (2.7)$$

where $H(x)$ is the prior entropy (initial uncertainty), and $p_x(m_k)$ is the prior probability defined on the certain voxel $x \in X$, which is approximated as $p_x(m_k) = p(m_k)$. The uncertainty after observation is determined by the conditional entropy as

$$H(x \mid m_i) = p_x(0 \mid m_i) H_x(0) + p_x(1 \mid m_i) H_x(1) \quad (2.8)$$

where $p_x(1|m_i) = f(x, m_i)$, $p_x(0|m_i) = 1 - p_x(1|m_i)$,

$$H_x(\delta) = \sum_{i=1}^{n_s} -p_x(m_k | \delta) \log p_x(m_k | \delta), \delta = 0, 1 \quad (2.9)$$

where

$$p_x(m_k | \delta) = \frac{p_x(m_i) p_x(\delta | m_i)}{\sum_{k=1}^{n_s} p_x(m_k) p_x(\delta | m_k)}, \delta = 0, 1 \quad (2.10)$$

The quality of viewpoint V_j is defined as the above conditional information gain, which measures the completeness and recognition rate of the scan obtained from V_j . The viewpoint with the highest information gain is selected as the NBV. This strategy tends to find as few viewpoints as possible to perform detailed scanning and recognition of γ .

2.3.1.4 Results Presentation

A series of experiments were conducted in this technical case, including simulations in virtual scenes and robots running in real scenes, to evaluate the effectiveness of this method. The evaluation is mainly designed around two questions about object-guided automatic scanning. First, how well can this method recognize objects in the scene? Second, how complete is the scan coverage of the object?

System Implementation

- (1) **Robot System:** This method operates on a custom robot platform, which has a six-degree-of-freedom joint arm equipped with a Microsoft Kinect RGB-D sensor, powered by the robot platform's onboard battery. The robot has a built-in computer running ROS (robot operating system), where ROS provides a package to enable standard robot behaviors, such as navigation and arm movements. Given a target viewpoint, the robot's position and pose information are computed, and the ROS package automatically plans the optimal collision-free smooth path.
- (2) **Dataset:** The virtual scene dataset is built on SUNCG [30] and ScanNet [31], where SUNCG includes 66 scenes synthesized by human simulation, and ScanNet includes 38 real scenes scanned by humans. The collection includes 104 scenes from 5 categories, including bedrooms, living rooms, kitchens, bathrooms, and offices. Both datasets provide real object segmentation and labels for the scenes.
- (3) **Benchmark:** To promote the quantitative evaluation of object-aware scene scanning, this method proposes a benchmark based on the virtual scene dataset,

called the object-aware scanning benchmark (OASC). However, please note that most scenes are not composed of objects from the 3D model database. This method evaluates the performance of object-aware scanning from multiple aspects, including object recognition, single-view object detection, object-level segmentation, object coverage, and object coverage quality (the detailed explanation of the benchmark evaluation results is not the focus of this section, for those interested, please refer to reference [2], detailed introduction is not provided here).

- (4) **Objectness Threshold:** When the objectness score of a post-segmented object is greater than 0.96, it is considered a complete object, and then the object is replaced by the most similar model in the database; when the objectness score of a post-segmented object is less than 0.05, it is considered noise in the scene and is filtered out from the scene surface T .
- (5) **Selection of Similar Model Set:** For the pre-segmented component $c \in C$, the most similar model is selected from the entire database M to construct $M(c)$, and it is used in post-segmentation to obtain the post-segmented object. For the post-segmented object $r \in R$, the most similar model is selected from the subset M that only contains complete models, because they provide complete information for the original 3D models in M^* .
- (6) **Floor and Walls:** The input scene is considered as a planar layout of floors, walls, and various furniture. Floors and walls can be easily identified using some heuristic plane fitting methods. They are removed from the scene surface T for object recognition and identification. This method uses the leading edge of the floor to guide the robot Ω to move to unexplored areas, preventing no objects from within visible distance. When there are no more objects in the scene surface T and no more leading edges on the floor, the scanning process terminates.

Experimental Results

- (1) **Virtual Scene Simulation:** Fig. 2.8 shows the results of the object-guided autonomous scanning reconstruction method in virtual simulation scenes. This method operates under the gazebo robot simulation environment [32] in ROS. The virtual scenes come from SUNCG and ScanNet. For each example, the top-down view of the recognized objects is visualized, where recognized



Fig. 2.8 Visual results of object-aware scanning for virtual simulation

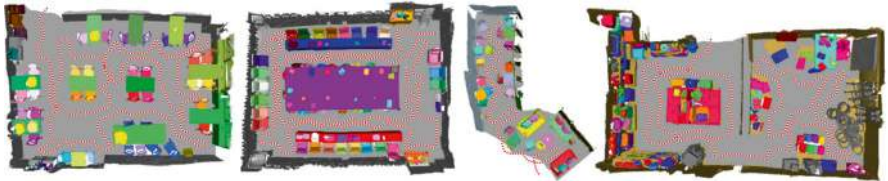


Fig. 2.9 Visual results of object-aware scanning for real running

Table 2.1 Running time (in minutes) of the object-aware approach and its various algorithmic components

Category	Total	Navigation	Segmentation	NBO	NBV
Bedroom (V)	47.8	24.1	20.1	2.0	1.6
Living room (V)	57.0	30.4	22.2	2.3	2.1
Kitchen (V)	37.5	16.2	17.6	2.0	1.7
Bathroom (V)	29.5	14.8	12.2	1.3	1.2
Office (V)	40.8	21.3	16.0	1.9	1.6
Meeting room (R)	101.4	62.3	32.4	3.6	3.1
Lounge (R)	78.5	47.9	25.4	2.9	2.3
Office (R)	94.7	56.9	30.3	4.2	3.3

- objects are shown in bright colors, unrecognized objects are shown in gray, and the robot’s navigation and scanning paths are also displayed. It can be seen that in this method, with the help of the object-based scene exploration method, the robot can autonomously scan and reconstruct unknown scenes along simple paths, and only one planning is needed to completely scan the entire scene, without the need for a second scan, i.e., one-time scanning.
- (2) **Real Robot Test:** Tests were conducted in four real-world unknown indoor rooms, including a cafe, a meeting room, a lounge, and a small shop, to simulate real-world scenes. Figure 2.9 shows the visual results of object-aware reconstruction in these scenes.
 - (3) **Running Time:** Table 2.1 records the online running time of this method, as well as the running time of each part of the algorithm, including navigation, segmentation, and the calculation of NBO and NBV. The time for the virtual scene (V) is the average of all scene times in each scene category. The scans of the three real scenes (R) were timed within the actual running time of the robot.

2.3.1.5 Chapter Summary

This technical case proposes a method for scene-autonomous exploration, reconstruction, and understanding guided by objects. The core of this method is a series of object-driven scene exploration strategies. First, a model-driven objectivity concept is defined and used to measure the similarity and integrity of pre-segmented components. Second, an objectivity-based segmentation is proposed to obtain a set of post-segmented objects. Next, an objectivity-based NBO strategy is used to

identify and recognize the OOI with the highest objectivity score and visual saliency, allowing the robot to start actively scanning the OOI, followed by calculating a small number of viewpoints through the NBV method to guide the robot's scanning, ensuring that the OOI is completely scanned. After the robot identifies the OOI, it can be replaced with the most similar 3D model retrieved from the database. Finally, move to the next OOI and start scanning it. This process is repeated until all objects in the scene have been scanned and reconstructed, thus achieving autonomous scanning and complete reconstruction of the scene.

2.3.2 Case Study: Design and Implementation of a System for Unknown 3D Scene Reconstruction

2.3.2.1 Task Description

In recent years, with the widespread use of RGB-D depth cameras and significant progress in scene reconstruction, high-quality scene scanning and high-density 3D reconstruction of indoor environments by mobile robots have attracted increasing attention from the robotics and graphics communities. In 3D scene reconstruction, humans can quickly construct the shape of objects in the scene relying on advanced visual intelligence. However, due to the complexity of vision problems, rich scene understanding information, large computation, etc., current robot vision systems find it hard to reach human cognitive abilities. The spatial relationships of objects in the scene are complex, occlusions can cause data loss, and the association between objects can lead to segmentation difficulties. Moreover, the complex and diverse appearance textures of objects, changes in lighting conditions can cause recognition difficulties, and complex geometric shapes can also affect modeling effects.

The reconstruction of unknown scenes is usually driven by the spatial information of the scene. Therefore, the key to the reconstruction of unknown 3D scenes lies in the implementation of automatic navigation by robots, thereby obtaining range scene information. High-quality scanning for scene reconstruction depends on the local geometric information of the visible surface, which requires the robot to move precisely to ensure complete and stable scanning, and to be able to obtain continuous frames, and reduce scanning and reconstruction errors. Balancing rapid scanning of the environment and high-quality scene reconstruction is the main challenge of autonomous scene reconstruction.

To enable robots to better understand unknown 3D scenes, this book proposes a method for unknown 3D scene reconstruction based on reconstruction navigation. The 3D reconstruction uses a voxel fusion-based method for online reconstruction, and the reconstructed scene has a constraining and guiding effect on the robot's walking and the next step of reconstruction scanning, with navigation carried out through a 2D field. It takes into account both the smoothness of local motion and the efficiency of exploration. The ultimate goal is to use cost-effective path navigation for high-quality reconstruction.



Chapter Knowledge Points

RGB-D Depth Camera: The RGB-D depth camera is a sensor that can simultaneously capture color RGB images and depth images, characterized by low energy consumption, low price, high-quality image information, and high real-time performance [4]. In recent years, significant breakthroughs have been made in the research of 3D scene reconstruction based on RGB-D cameras. Conventional methods generally use fast image processing technology for depth image registration, which can construct dense 3D scenes with color information, directly applied to tasks such as robot navigation [5]. Kinect is an RGB-D sensor that not only has fast collection speed and high accuracy but also is cheaper. These characteristics have quickly made it applicable to 3D reconstruction tasks. With the widespread use of Kinect depth cameras, it has also played a promoting role in robot navigation. Using Kinect depth cameras for 3D scene modeling and obtaining different types of representations of 3D models has been a hot research topic in recent years [4–6].

ROS: ROS is an acronym for robot operating system. ROS is a highly flexible software architecture for writing robot software programs. The prototype of ROS comes from the stanford artificial intelligence robot (STAIR) and personal robotics (PR) projects at Stanford University.

TSDF: Truncated signed distance function is a common method for calculating implicit surfaces in 3D reconstruction. The famous KinectFusion uses TSDF to construct spatial voxels, by calculating the value of each voxel, and then using the previously mentioned Marching Cube to extract the surface.

SDF and TSDF: TSDF should actually be called T-SDF, where SDF is a signed distance field, and T represents truncation. TSDF has much less computation than SDF. T-SDF fusion involves adding all point cloud data into a voxel, calculating the SDF value in the related voxel, then constructing the isosurface, and finally obtaining the mesh model. This method of expressing isosurfaces can be classified as implicit representation.

2.3.2.2 Background Technology

One of the core and key technologies of mobile robots is navigation technology. The path planning of mobile robots is to allow robots to autonomously plan a path suitable for the scene based on the information of the indoor scene, or guided by the external environment obtained by sensors.

A very direct method of scene scanning is to manually control the robot to scan the scene. The result is that the scanned data is uncertain and is controlled by human subjective will. The definition of robot path planning refers to the optimal path planning problem, that is, based on one or several optimization criteria (such as minimum cost, shortest movement path, and shortest movement time), find an optimal path that can avoid obstacles in the scene to be scanned and reconstructed, and can

pass through the start and end points. Common methods include the Dijkstra algorithm, search area algorithm, and open list algorithm.

However, these are methods when the scene structure is known. This article is dedicated to the exploration of unknown scenes, which is based on 3D scene reconstruction path planning. The main technical challenge is to implement robot path planning, thereby solving the balance problem between local reconstruction quality and global efficiency. Below is a brief introduction to several common path-planning algorithms and the methods used in this article.

Dijkstra Algorithm

The Dijkstra algorithm was proposed by E.W. Dijkstra around 1960 and is the most classic single-source shortest path algorithm. In the application of robot navigation, it can find the shortest path from one node to all other nodes in the robot's moving route. The method is to take the robot's starting position as the origin, adopt a gradually expanding undirected graph until it includes the point to be moved to, and form the path map of the entire scene by calculating the edges formed between the nodes and the relationship of the edge weights.

Overall description of the Dijkstra algorithm: In the undirected graph $G = (V, E)$, assuming the length of edge $E[i]$ is $W[i]$, the goal is to find the shortest path from vertex $V[0]$ to all other nodes.

AMCL Algorithm

The AMCL (adaptive Monte Carlo localization) algorithm is an adaptive Monte Carlo localization algorithm that uses a particle filter to track the robot's position in a known 3D scene. AMCL is a probabilistic localization system for robots in two dimensions. By randomly scattering different particles in the map and finding the path that can be moved according to their collision feedback, the algorithm represents the robot's pose information in the form of particles. AMCL uses a finite number of particles to represent the probability distribution of robot movement. If the number of particles is sufficient, a probability distribution function almost identical to the actual robot can be obtained. The disadvantage of the AMCL algorithm is that it requires a large amount of computation, resulting in low real-time performance of the algorithm.

Field-Based Path Navigation

The path planning part of this paper uses a 2D direction field to guide the robot's movement and then forms a smooth path through particle advection. The field of the visualizable 2D image includes potential field, gradient field, and tensor field [33].

The concept of the tensor field mentioned by PALACIOS et al. [33] is briefly introduced below. The vector field in a 2D image can be visualized as: A collection of arrows with a given magnitude and direction, each arrow connected to a point on the plane. Vector fields are often used to simulate, for example, the direction and speed of a fluid in space, or the strength and direction of a force at a point, as it changes from one point to another. A tensor field on a 2D region D is a smooth tensor-valued function T , which is associated with a second-order tensor T formula (2.11) for each point P belonging to D . The tensor T_{ij} is symmetric if and only if $T_{ij} = T_{ji}$.

$$T(P) = \begin{pmatrix} T_{11}(P) & T_{12}(P) \\ T_{21}(P) & T_{22}(P) \end{pmatrix} \quad (2.11)$$

The symmetric tensor T can be uniquely decomposed into its isotropic part S and anisotropic (deviation) part A as shown in formula (2.12). The deviation tensor field $A(P)$ is equivalent to two orthogonal eigenvector fields $E_1(p) = \mu(p) \cdot e_1(p)$ and $E_2(p) = \mu(p) \cdot e_2(p)$, where $e_1(P)$ and $e_2(P)$ are unit eigenvectors corresponding to eigenvalues μ and $-\mu$, respectively, and E_1 and E_2 are the principal and minor eigenvector fields of A . A point P_0 is a degenerate point of the tensor T if and only if $A(P_0) = 0$.

$$T = S + A = \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mu \begin{pmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{pmatrix} \quad (2.12)$$

The principal and minor eigenvectors of T are orthogonal to each other. In the experiment, the robot's movement is directed by the principal eigenvector. The main feature of the symmetric tensor field is that its direction (sign) is quite ambiguous everywhere, making it equivalent to a line field that does not distinguish between forward and backward. This avoids the problem of calculating the vector direction of the vector field under geometric constraints. The degenerate points of the tensor field usually have the same function as the singular points of the vector field.

Figure 2.10 shows the effect diagrams of potential fields, gradient fields, and tensor fields, where the green dots represent singularities, which can be intuitively considered as turning points for the robot's walking. The potential field is generated

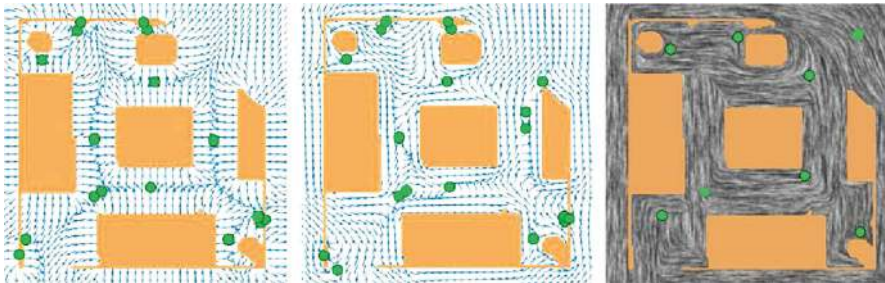


Fig. 2.10 Degenerate points of potential fields, gradient fields, and tensor fields

by the surface normal, while the gradient field is constrained by the surface tangent. Due to the inconsistency in direction, these two vector fields are troubled by crowded singularities (green dots), and if the singularities are too dense, they will hinder the issuance of walking instructions for the robot. In contrast, the tensor field contains far fewer singularities, and fewer singularities can lead to smooth path advection (particle advection). Moreover, the topological skeleton of the field is composed of all degenerate points and the boundary points connecting them, which can be regarded as a routing map. Using this global structure, global path planning for efficient scene scanning can be achieved.

2.3.2.3 Technical Implementation

This technical case aims to implement an unknown indoor scene reconstruction task based on reconstruction navigation, with the ultimate goal of enabling the robot to automatically complete the reconstruction task of the indoor scene location. Therefore, the implementation part is divided into two modules: the first module introduces the real-time reconstruction method; and the second module introduces the reconstruction navigation method.

Since it is real-time modeling, the two parts are carried out simultaneously. During the reconstruction of the scene under the current camera view, the scene (mesh) reconstructed and projected onto the 2D plane will issue instructions for the robot’s scanning path, guiding the robot to perform the next unknown scanning reconstruction. The reconstruction results will continue to navigate the robot’s movement, guiding the entire scene to be scanned and reconstructed until completion. The system pipeline is shown in Fig. 2.11.

3D Scene Reconstruction Based on Voxel Fusion

When performing traditional reconstruction methods based on depth cameras, the reconstruction space is refined into voxels of the same size, which consumes a lot of memory [23], making it difficult to reconstruct large scenes, and large-scale voxel

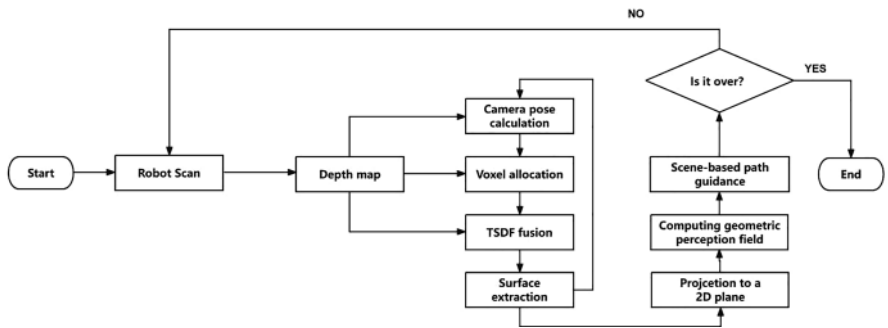


Fig. 2.11 System pipeline

updates will consume GPU resources. The voxel fusion (volumetric fusion) improvement method proposed in this technical case only divides voxels on the scene surface under the current camera view, thus saving memory. Although it reduces the data for effective calculation, it will lead to sparse effective data. To solve this problem, the sparse data can be transformed into dense data. The idea is to store the grid to be calculated in a hash table according to the x , y , and z coordinate values of the grid in the real world and to store the divided voxel blocks in the form of a hash table for various operations.

The goal of this technical case is to establish a real-time reconstruction system, using a spatial hashing method for scalable volume reconstruction, integrating the information of the current frame scanned by the camera into the final model, and the reconstruction result is in the form of a mesh. Because there is uncertainty when scanning the scene under unknown conditions, the hashing method must support dynamic allocation and updates, while minimizing and resolving potential hash input conflicts to the greatest extent. The hash table stores and updates T-SDF sparsely and efficiently. The voxel blocks between the host and the GPU are streamed, and the isosurface is effectively extracted from the data structure using standard ray projection or polygonization operations for rendering and camera pose estimation.

The data structure of the hash table is used to store voxel sub-blocks containing T-SDF values. Each occupied position in the hash table corresponds to an allocated voxel. Each voxel stores T-SDF, weight, and additional RGB color values. The hash table is unstructured, that is, adjacent voxel blocks are not stored adjacently in the hash table but are located in different parts of the hash table. The hash function allows the voxel block to be effectively found using the specified integer world coordinates. At the same time, the hash function aims to minimize the number of collisions and ensure that there are no duplicate items in the table.

The reconstruction process based on voxel fusion is shown in Fig. 2.12. Given a new input depth map, new voxel blocks are first allocated according to the input depth map and inserted into the hash table, only allocating occupied voxels and not allocating blank space voxels. Then, each allocated voxel block is scanned, and the T-SDF, color, and weight of each voxel-containing block are updated based on the input depth and color samples. In addition, voxel blocks that are far from the isosurface and have no weight are collected, which involves releasing allocated GPU memory and removing voxel block entries from the hash table. These steps ensure that the data structure maintains its sparse characteristics over time.

After T-SDF fusion, ray casting is performed on the implicit surface of the current estimated camera pose to extract the reconstructed surface under the current camera view. This extracted surface is used as input for camera pose estimation:

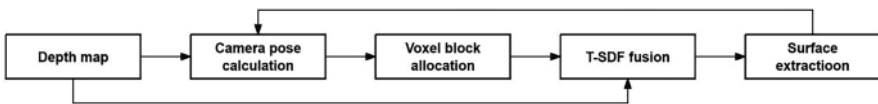


Fig. 2.12 Online reconstruction process

given the next input depth map, a projection point plane ICP is executed to estimate the new 6DoF camera pose, ensuring that camera pose estimation is performed from frame to model. Finally, the algorithm performs bidirectional streaming between the GPU and the host. Hash entries and related voxel blocks are streamed to the host after world coordinate estimation is complete and they leave the camera's conical view. When revisiting areas, previously streamed-out voxel blocks can also be streamed back to the GPU data structure. The reconstructed voxel blocks can move bidirectionally between the CPU and GPU.

Reconstruction Navigation Based on 2D Tensor Fields

The autonomous reconstruction system consists of a mobile robot, which uses its top-mounted RGB-D camera and the voxel fusion method proposed in the previous section for real-time reconstruction, while also using OctoMap [34] to store complex spatial occupancy information.

The main idea is to use a 2D direction field to guide robot movement. During online scanning and reconstruction, the tensor field is updated in real time, making edge scanning and reconstruction possible. The 2D direction field is related to the already reconstructed 3D scene, which is projected onto the ground from the reconstructed 3D scene, and the method of Palacios et al. [33] is used to extract a set of 2D tangents along the projected boundary for calculation and field update. The robot path is formed by particle convection on the tangential direction field, essentially avoiding obstacles. A spatiotemporal optimization of the tensor field is proposed through strong time coherence, and interpolation is used to ensure smooth robot paths when translating particles on time-varying fields. After obtaining the robot path for the next time interval, it is necessary to calculate the camera trajectory flow of the scene scan along this path segment.

The reconstruction navigation process based on the 2D tensor field is shown in Fig. 2.13.

Projection to a 2D Plane

OctoMap [34] is an octree compression method that can keep 3D models compact. The provided framework is available as an open-source C++ library. Storing the currently reconstructed 3D model of the frame using OctoMap not only maintains the original shape of the model but also significantly reduces the consumed GPU memory.

The field of the 2D image can be visualized. During the simulation experiment, a simple indoor scene was simulated on the Gazebo simulator, as shown in Fig. 2.14.

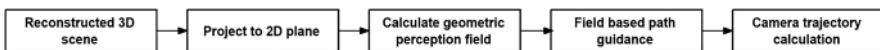


Fig. 2.13 Reconstruction navigation algorithm process

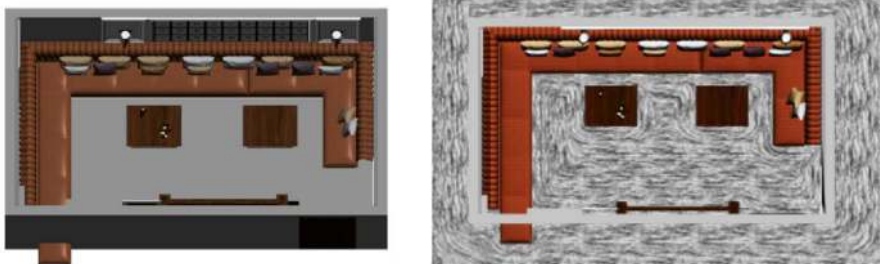


Fig. 2.14 3D scene and 2D tensor field

The scene was projected and the tensor field was visualized. Intuitively, if there are different objects in the image, the shapes of the fields around the surfaces of these objects can be seen as the tracks of a playground, with straight lines around the objects that can be seen as the paths of robot movement.

Geometric-Aware Tensor Field Update

- (1) Generation of keyframe field: While using voxel fusion for frame reconstruction, the reconstructed 3D volume is efficiently stored using OctoMap, then projected onto the ground plane, and point sampling is performed on the projected image boundaries located on the known surface. Then, tensors are assigned to each sampled constraint point, with their main direction tangent to the corresponding surface. These 2D tensors are used as constraints to calculate the keyframe tensor field. This method can only project 3D surfaces below the height of the robot's head or the TurtleBot's camera onto the plane. The sampling distance is set to $L = 0.3$ m, and then a regular tensor field is set for each sampled point, with the eigenvector parallel to the tangent of the image boundary at that point.

The final field is composed of the tensor fields of all constraint points, which can be equivalent to a mathematical problem: find a function T , which is a mapping from a point in 2D space to the tensor T_i , given some sampled instances $[P, T_i(P)]$. The Gaussian radial basis function (RBF) is usually used to approximate this function [33]. The calculated function is as shown in Eq. (2.13):

$$T(P) = \sum_i e^{-\frac{\|P - P_i\|^2}{\sigma^2}} T_i(P) \quad (2.13)$$

where T_i is the field calculated around the constraint point P_i , the Gaussian action width σ is used to control the influence range of the field, which is mainly used for local path guidance, so a smaller value of $2L = 0.6$ m is chosen.

- (2) Time interpolation method: At each time step, the current frame field T is calculated based on the latest reconstructed scene. If waiting until the current time

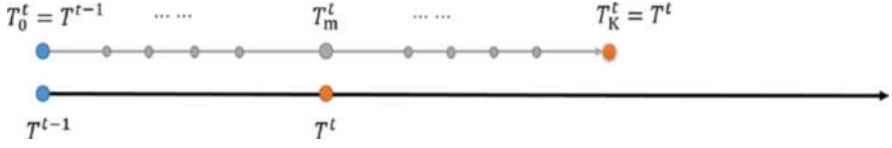


Fig. 2.15 Time interpolation method for obtaining the tensor field at time T^t

period, time consistency cannot be ensured, which may lead to robot route deviation or even collision. However, it can be smoothly transitioned by interpolating the previous keyframe T^{t-1} into K substeps, which can satisfy the real-time performance of the system.

A time-coherent sequence $(T_0^t = T^{t-1}, \dots, T_K^t = T^t)$ is generated using linear interpolation, and then the tensor field is evolved from the middle frame $[T_m^t (m = K/2)]$ until $T_K^t = T^t$, and then the next frame tensor field is solved. The interpolation process is shown in Fig. 2.15, with K substeps (the small gray points in Fig. 2.15 are interpolated, and the field is evolved from the middle frame T_m^t).

Path Guidance Based on the Field

- (1) Particle advection to generate paths: As the known 3D surface is gradually reconstructed, the tensor field projected onto the 2D plane also changes with time. The particle advection method can release particles in the changing field, track the path of the particles in the field, and thus issue commands for the robot's route, guiding the robot to complete the scene scan.

After calculating the field that changes with time, the robot's motion path from the current position is defined by particle advection, with the main eigenvector of the tensor located in the local area around the current robot position. This is achieved by making the main eigenvector of the tensor field close to the direction in front of the robot (the angle between the directions should be less than 90°). Once the field direction is determined, the field T can be turned into a scalar field $V(t)$, where particle advection can be represented as a differential equation, as shown in Eq. (2.14). The initial value is $(p_0; t_0)$, defining the path from position p_0 and time t_0 .

$$p(t) = p_0 + \int_0^t V(p(s); t_0 + t) ds \quad (2.14)$$

In the experiment, the path of four substep times is calculated, and the path advection is restarted when the robot moves to the next position. Because the field changes smoothly, the paths between coherent substeps are smoothly connected. As shown in Fig. 2.16, based on the smooth transition field from T_m^t to T^t , a smooth robot path is generated in each substep by moving a particle on the current field.

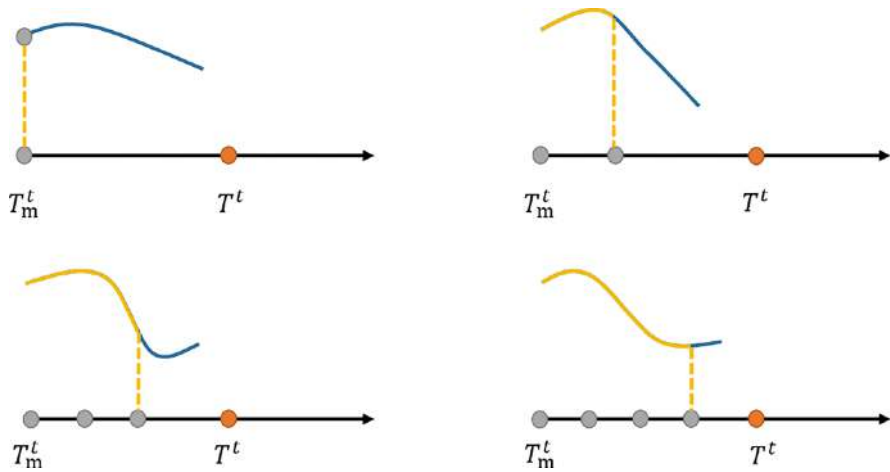


Fig. 2.16 Generation of smooth paths

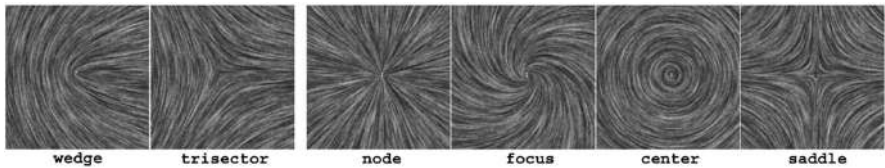


Fig. 2.17 Types of field degenerate points

- (2) Path selection at turning points: In more complex indoor scenes, robots may need to turn during their movement, and the degenerate points of the tensor field intuitively meet this requirement. During the scanning and reconstruction process, encountering new objects can cause the surrounding field to become irregular, and the robot's path is no longer a straight line-like path. Special handling is needed at these degenerate points. Palacios et al. [33] proposed that degenerate points include wedge points, tri-sector points, nodes, focus points, center points, and saddle points, as shown in Fig. 2.17. In the application to 3D scene reconstruction, most are wedge points and tri-sector points.
- (a) At the wedge point. Near the wedge point as shown on the left side of Fig. 2.18, according to the characteristics of the 2D field and the influence of path advection, the robot's path will make a sharp turn or U-turn. To avoid camera shaking caused by this sudden turn, detect the wedge point closest to the robot's path and reduce the robot's moving speed until it reaches the wedge point. At the wedge point, the robot will stop moving and replan the path, allowing the robot to move smoothly within the time of the two frames.

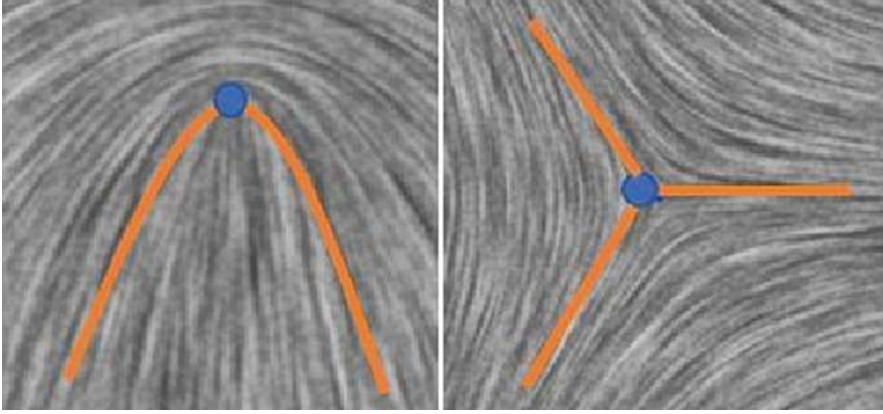


Fig. 2.18 Path at degenerate points

- (b) At the tri-sector point. When the robot reaches the tri-sector point, it can choose to go straight or take the path of the wedge point in (a), as shown on the right side of Fig. 2.18. This ambiguous problem is solved by choosing a better branch based on reconstruction uncertainty. Specifically, choose the branch that can explore more unknown or uncertain areas. After determining the branch, execute the same path and trajectory as the wedge point.
- (3) Formation of topological structure: The entire scene to be reconstructed can be regarded as a local area network, which can be covered by several routers. These routers form the topological structure of the entire local area network. Projecting the entire 3D scene to be scanned onto a 2D tensor field on a plane can be regarded as a problem in generating a topological structure.

Locally smooth paths cannot guarantee efficient and complete coverage of the entire scene. Global navigation needs to be carried out according to the topological skeleton of the current tensor field. Given any tensor field, its topological skeleton can be regarded as an undirected graph, all degenerate points are nodes of the graph, and the connections between them are boundaries. The main focus is on which branch to choose at the tri-sector point. The tri-sector points in the topological skeleton serve as “gateways in the global path routing, the robot performs branch selection, and the branch with the smallest cost is selected by calculating the minimum cost generation tree from the topological graph.” In Fig. 2.19, the weight of each edge is defined as formula (2.15):

$$w(e) = \left(\frac{1}{l(e)} \int_e^{\infty} I(e, p) dp \right)^{-1} \quad (2.15)$$

where $l(e)$ is the path length of edge e , $I(e, p)$ is the information gain at point p on edge e , which is defined in reference [7] as how many unknown areas can be

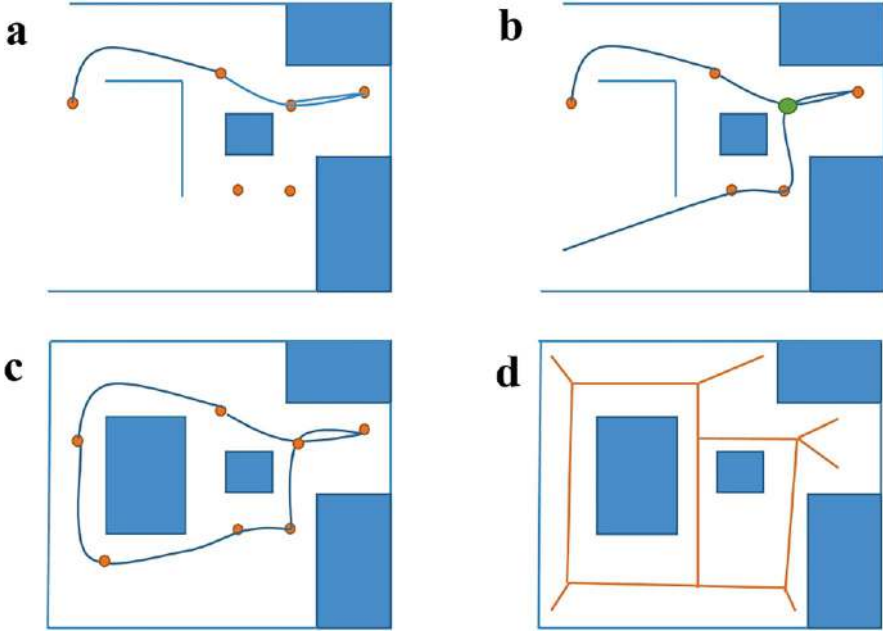


Fig. 2.19 Navigation based on topological structure

observed in the current view in the current reconstruction scene, and the scanning distance of the view represents the information gain of a specific viewpoint. By choosing the branch of the path with the smallest cost, the scene can be scanned more efficiently.

A notable feature of the path based on the topological structure is that it works well for partially reconstructed scenes. This is due to two advantageous features of the tensor field. First, any tensor field calculated with any geometric constraints can well define the topological graph. Second, the minimization of the number of singularities in the tensor field prevents the appearance of irrelevant singularities in free space, providing a simple and meaningful topological map for local scenes. The minimum spanning tree under the current position can be conveniently formed through the coordinates of the singularities on the 2D plane and the distances between points. As shown in Fig. 2.19, the topological skeleton of the tensor field is calculated for the partially reconstructed scene (a) and used to guide the robot's scanning. When the robot (in green) reaches the trisection point, the minimum cost generation tree is generated from the topological map to enable branch selection (b). When the reconstruction is complete, the generated topological structure (c) is basically the same as the edge coverage of the scene (d).

- (4) Collision handling: The tensor field is formed by the constraints of surface tangents, which essentially avoids obstacles in the path advection. However, there is a situation that needs special handling. If the robot's initial position is close

to the wall, the robot will be close to the wall, because it is walking in the direction of the surface tangent at this time. Due to the non-negligible volume of the robot, this may cause a collision. Therefore, once it is detected that the robot is too close to a known wall (less than 0.3 m), the robot will be made to move away from the wall in the direction of its normal.

- (5) Termination condition: The termination of autonomous scanning of unknown 3D scenes based on the topological structure of the field and the definition of information gain is defined as: if the expected information gain of all accessible edges in the topological map is below the threshold, set to 0.05 [7] in the experiment. Based on the robot's body contact test and the reconstructed scene, the accessibility of the edge is determined by checking the physical accessibility of the sampling points along the edge. The topological skeleton of the field roughly reflects the topological structure of the boundary of the unknown 3D scene.

Calculation of Camera Trajectory

After obtaining the robot's path for the next time interval, it is necessary to calculate the camera trajectory for scanning the scene along this path segment. The goal is to calculate a smooth 6DoF camera trajectory that meets the following requirements. Drawing on the method of reference [32], in order to effectively scan the scene, the camera view along the trajectory should obtain the maximum coverage of unknown or uncertain areas based on the current reconstruction status. A good viewpoint should observe the target area from a close distance (within the effective scanning range) and be as perpendicular to the target surface as possible. In order to facilitate frame-to-frame registration during the online reconstruction process, the linear speed and angular speed of the camera should be kept as constant as possible, not exceeding the maximum threshold. During operation, the smoothness of the robot's motion can be reflected by monitoring the change in the camera's angular acceleration rad/s^2 .

System Optimization

- (1) Improvement of online reconstruction performance. Although most of the calculations are hash operations in the voxel blocks under the current view of the camera, the surface of the part that has been reconstructed still needs to be maintained, which is a great demand for the memory and performance of the GPU. In the experiment, a two-way stream (steam) scheme of GPU and CPU is used to reduce the execution time of the algorithm. Streaming can be seen as a data processing engine that can handle infinite data.

Voxel blocks that have been reconstructed and are not within the current view of the camera need to be moved from the GPU to the CPU, and voxel blocks outside a circle with a radius of 4 m with the current position of the camera as the origin are moved to the CPU for storage.

GPU-CPU: First, access the hash table in parallel, mark the voxel blocks in the active area, and copy the hash entries to one buffer, and the voxel blocks corresponding to the hash entries to another buffer. Delete the original hash entries and corresponding voxel blocks, and then copy the hash entries and voxel blocks in the buffer to memory.

CPU-GPU: Due to the movement of the camera, new voxel blocks within the current camera view in the active area have been reconstructed and stored in memory. These blocks need to be moved from memory to the GPU and new units allocated within the GPU.

An important consideration in streaming is to ensure that voxel blocks are not duplicated on the host or GPU, which could lead to potential memory leaks.

- (2) Automatic optimization of the field. Another major advantage of the tensor field is that it allows operations on degenerate points. The above reconstruction navigation method is mainly designed for degenerate points, which may result in several situations as shown in Fig. 2.20: (a) The degenerate point in the figure is in a rather sharp position, the robot will take a sudden turn, which can lead to jittery movement. (b) The situation in the figure shows a concave area that can be covered by the camera's view, but the robot will still go in for scanning. (c) The degenerate point that appears in the figure is a trisect point, the robot will make a path choice, and the sudden change in the field may cause the robot to collide with obstacles.

To avoid the above situations, some modifications need to be made to the position of the degenerate points, i.e., moving some degenerate points or deleting some redundant points. The basic idea is to convert the tensor field into a vector field according to the definition of field T , as follows: $V = \alpha(T) = \mu(\cos 2\theta, \sin 2\theta)^T$, using the modification method in reference [33] to transform V into V' , and then back to tensor field $T' = \alpha^{-1}(V')$. The optimization part needs to be automatically modified after calculating the field of each keyframe.

- (1) Moving degenerate points: For wedge points, if the angle between the two dividing lines is less than 90° , it can be seen as a sharp corner, at this time the point can be moved inward to a corner, making the robot's turn smoother, as

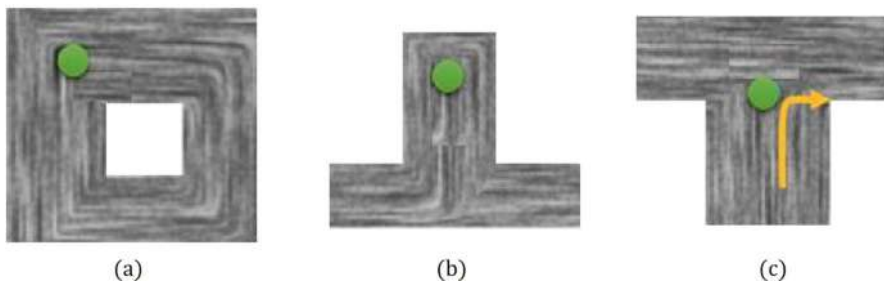


Fig. 2.20 Degenerate points to be moved

shown in Fig. 2.21. Figure 2.21a is what it looks like after moving. Generally, the wedge point is the end point of the topological structure. If there is no more scene information after this point, the robot does not need to probe deeper, the degenerate point can be moved outward a short distance, eliminating unnecessary movement at the end, see Fig. 2.21b. The third situation is the handling of trisect points. Since the robot may move along the dividing line to the trisect point, it is necessary to ensure that the robot will not collide with obstacles when it reaches the trisect point or turns into a branch, move the trisect point away from the obstacle, until no collision is detected along the trisect point or the sampling point on the trisect line, see Fig. 2.21c.

- (2) Canceling pairs of degenerate points: Due to the complexity of the relative positions of objects in the scene, degenerate points in the field may cluster together or be close to each other, as shown in Fig. 2.22. This can confuse the issuance of movement instructions for the robot. If unnecessary degenerate points can be successfully eliminated, a geometrically smoother and topologically simpler field will be produced. Nieser et al. [35] proposed that a pair of degenerate points can only be offset if they are close enough, thus minimizing topological changes in the field due to offsetting. Similarly, pairs of degenerate points on the topological skeleton of the field that can be eliminated can be detected by calculating the shortest distance between them.

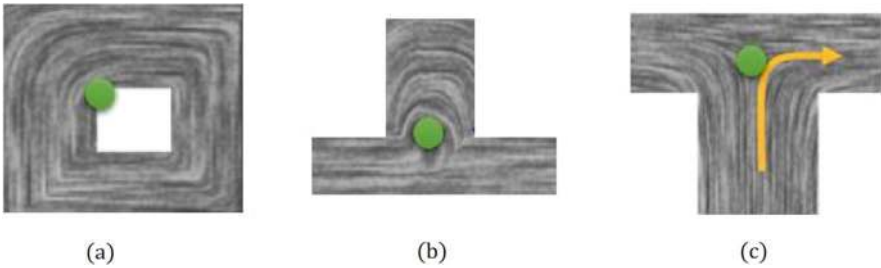


Fig. 2.21 Moving degenerate points

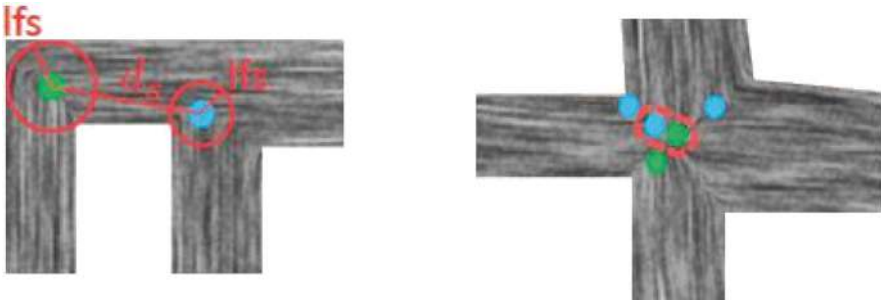


Fig. 2.22 Judging whether pairs of degenerate points can be canceled

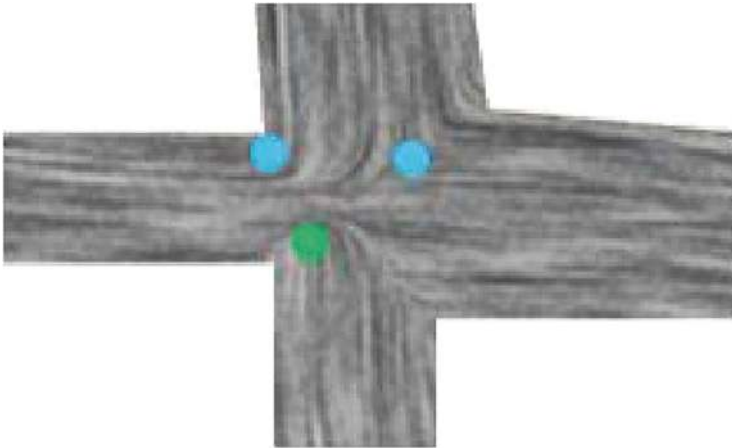


Fig. 2.23 Field after canceling pairs of points

In order to retain pairs of degenerate points that have a significant impact on the topological structure, the pair of points (pW, pT) can be canceled [33] if they satisfy Eq. (2.16):

$$d_s(pW, pT) < lfs(pW) + lfs(pT) \quad (2.16)$$

Here, d_s is the shortest distance between two points, and lfs is the radius of the degenerate point's neighborhood. If a pair of points satisfies the above conditions, they will be deleted. Since the field changes over time, a new field will be immediately updated, which will not have a significant impact on the robot's walking instructions. In Fig. 2.22, the pair of points on the left cannot be canceled because they do not satisfy Eq. (2.16), while the pair on the right can be canceled. The field after the cancellation of the pair of points is shown in Fig. 2.23.

2.3.2.4 Results Display

Online Reconstruction Experiment Section

Through the voxel fusion-based 3D reconstruction method, the graphics card needs to keep up with the frame rate of RGB-D (30fps) for reconstruction. This allows for the selection of different resolution voxels (2 mm, 4 mm, or 8 mm) for high-quality reconstruction during implicit surface fusion.

The experiment was conducted in a laboratory setting for 3D scene reconstruction testing, focusing mainly on the quality of the reconstruction. Common office equipment such as chairs and computers were scanned, and the scanning results are shown in Fig. 2.24. The voxel resolution is 8 mm, and both RGB and RGB-D images are displayed on the output screen. The reconstruction results for finer

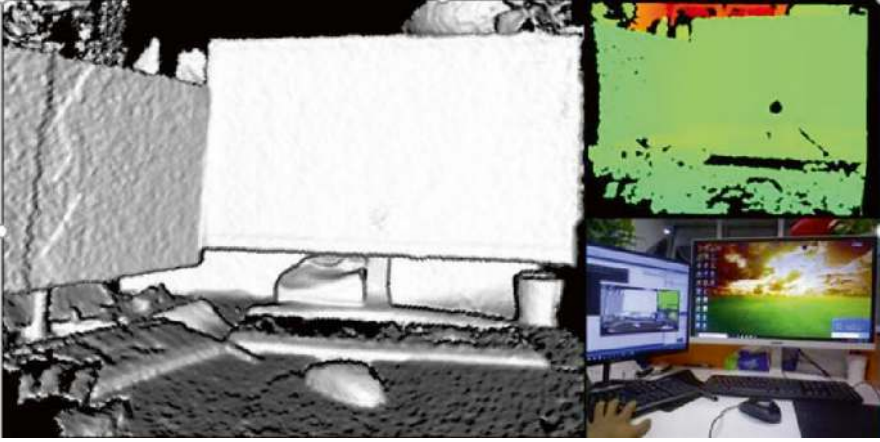


Fig. 2.24 Reconstruction results

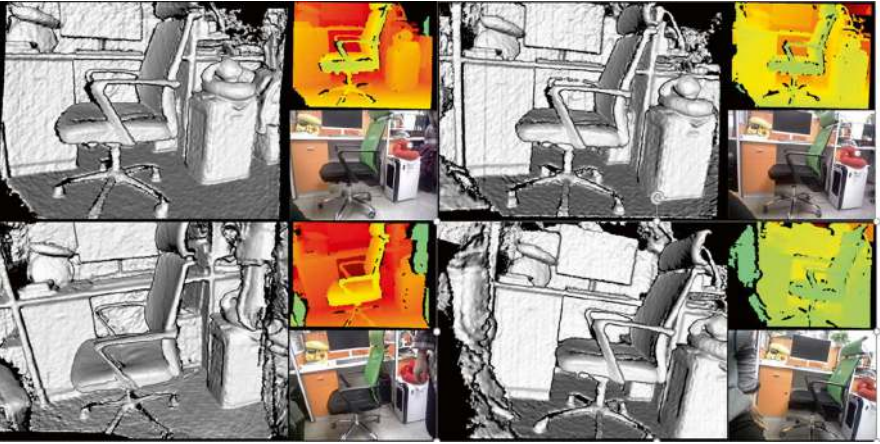


Fig. 2.25 Comparison results of different voxel resolutions

details, such as keyboards and mice, are also good, and the frame rate of scanning in real scenes can reach about 50 per second.

For different resolution voxels, comparative experiments were conducted with 2 mm (Fig. 2.25 top left), 4 mm (Fig. 2.25 top right), 8 mm (Fig. 2.25 bottom left), and 10 mm (Fig. 2.25 bottom right) voxel resolutions. The experimental object was a chair model, and the RGB information was not updated (output in gray mesh form). The comparison effect is shown in Fig. 2.25.

The experimental results show that the reconstruction results are not directly affected by the voxel resolution. The voxel resolution is related to the memory required for the code to run. Choosing different voxel resolutions according to different scene sizes makes the system more flexible. The experimental results also

show that the degree of detail in the reconstruction is related to the current depth sensor’s sensing technology. As the precision of the RGB-D camera increases, the experimental results will also greatly improve.

Comparison Experiment of Reconstruction Methods

In addition to the voxel fusion-based 3D scene reconstruction method, a feature descriptor-matching reconstruction method has also been attempted. After obtaining the segmented point cloud data through point cloud fusion and cooperative segmentation based on the scanned RGB-D, each object is retrieved, matched, and replaced with the database (ShapeNet) using feature description to achieve the purpose of scene reconstruction. The models in the database are preprocessed. The original plan’s process is shown in Fig. 2.26.

Since this method processes the fused point cloud, it results in a relatively large time complexity of the algorithm (a lot of time is spent on keypoint extraction and registration steps), making the system not very real-time, and therefore unable to complete the reconstruction navigation task. However, the reconstructed results are obtained by retrieving objects from a large database of models, resulting in a high degree of refinement in the reconstructed models, which is an advantage that the voxel fusion-based method does not have. The comparison between the reconstruction results based on voxel fusion (with RGB information updated) and those based on feature descriptor matching is shown in Fig. 2.27.

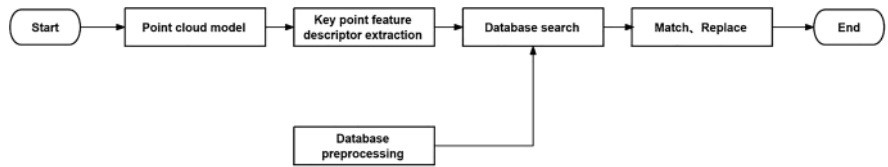


Fig. 2.26 Reconstruction method based on feature descriptor matching

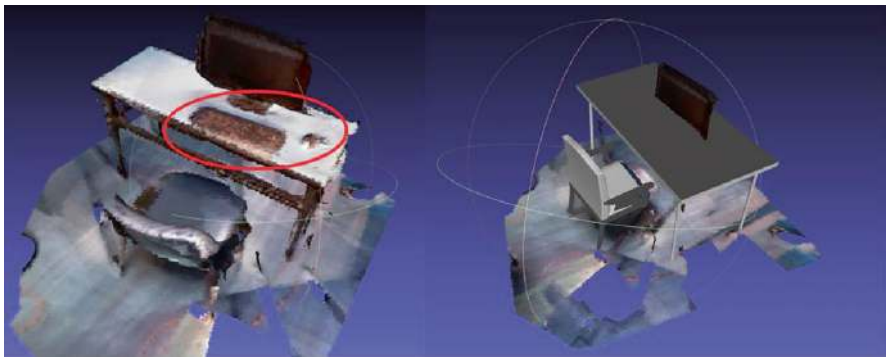


Fig. 2.27 Comparison results



Fig. 2.28 Indoor scene

It can be seen that the reconstruction method based on voxel fusion can reconstruct small objects in the scene, such as the keyboard in the red circle in the picture, but it does not consider the semantic information of various objects in the scene. On the contrary, the reconstruction algorithm based on feature descriptor matching is more refined, which is also at the cost of sacrificing the running time of the algorithm.

Path Navigation Experiment Part

In order to verify whether the algorithm mentioned in this chapter is feasible, experiments were first conducted in the gazebo simulation environment based on TurtleBot. Simulations were conducted in 40 simulated indoor scenes, with an area of 30–90 m², an average of 6 types of objects, and a maximum of 15 types. As shown in Fig. 2.28, four bookshelves, one table, and one coffee table were placed in the indoor scene. During the implicit surface update and surface extraction, the RGB information was not updated, that is, the preset color information (yellow, not gray, overlapping with the field color) was output. The robot's starting position was placed on the wall in the upper left corner, the time required for scene reconstruction was 155 s, and the length of the walking path was 47 m. For the 40 simulation scenes, due to the different complexity of the scene construction, the average reconstruction time was measured to be about 3 min.

The reconstruction result is shown in Fig. 2.29.

The quality of the reconstruction is judged by the root mean square error (RMSE) based on ICP registration. The two inputs of ICP matching are the current frame depth image and the depth image after the implicit surface update and ray projection according to the previous frame camera pose, which ensures that the camera pose estimation is frame-to-model. As shown in equation (2.17):

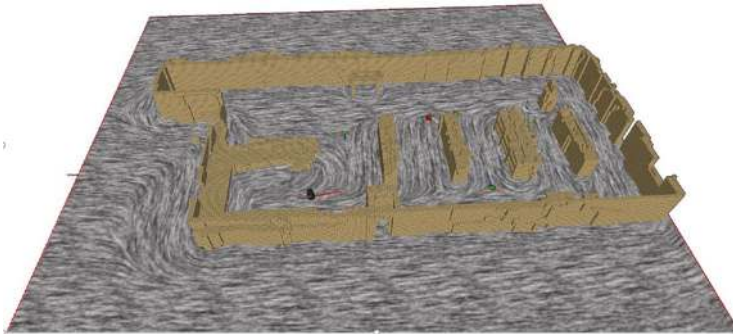


Fig. 2.29 Reconstruction result

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\text{observed}_i - \text{observed}_i)^2} \quad (2.17)$$

where $(\text{observed}_i - \text{observed}_i)^2$ is the total error between all corresponding points in the previous and next frames, and N is the total time of reconstruction.

In order to verify whether the tensor field update method using time interpolation (time step 4 s, step number five steps) is adopted, a comparative experiment without interpolation method (update time interval is 4 s) was conducted. The average RMSE of the interpolation method is 0.4 cm, and the average RMSE without the interpolation method is 0.8 cm. The experiment shows that the use of time interpolation method for tensor field update has a significant improvement on the quality of reconstruction.

2.3.2.5 Summary

This book introduces a method for reconstructing unknown 3D scenes. The efficiency and real-time performance of existing methods such as database retrieval and feature descriptor matching are not very high, and they cannot complete the reconstruction navigation step. The unknown 3D scene reconstruction system proposed in this book can issue instructions to the robot's movement direction through the information of the currently reconstructed 3D scene, navigate the robot to continue to scan and reconstruct, and the reconstruction results will continue to navigate the robot's movement until the entire scene is reconstructed.

The system uses a spatial hashing method for scalable volume reconstruction. The dynamic allocation and update of the hashing method can cope with the uncertainty of the scanning scene, and the memory occupied by the hash table is very low. After T-SDF fusion, the standard ray projection is used to extract the isosurface from the data structure. The reconstructed surface is projected onto a 2D plane, and the robot's walking route is navigated through the change of the tensor field. In terms of improving system performance, the streaming transmission of voxel

blocks between the host and the GPU is adopted, allowing for larger scene reconstruction.

The system was run in laboratory scenes and simulation environments and compared with existing reconstruction methods, with the following conclusions:

- (1) The voxel fusion method only divides voxels on the scene surface under the current camera perspective, thus saving memory. The divided voxel blocks are stored in the form of a hash table for various operations.
- (2) The reconstruction method based on voxel fusion significantly shortens the required time compared to the reconstruction method based on feature descriptor matching.
- (3) Path guidance based on a 2D field can generate correct robot scanning routes.
- (4) The quality of reconstruction is judged by the root mean squared error (RMSE) based on ICP registration. The error of tensor field updating using time interpolation is significantly lower than that without time interpolation.

The system also has some limitations that need to be improved in the future: During the initial stage of scanning, the reconstructed scene is not very useful for generating the robot's walking instructions. If the robot moves too fast, this may lead to collisions. The system may encounter loop closure problems during the reconstruction process, at which point ICP registration will fail. The reconstruction and navigation methods are geometric and do not consider the semantic information of the objects in the scene, lacking the function of robot and scene interaction.

2.3.3 Case Study: Semantic Scene Completion-Based Scene Exploration in Visual Semantic Navigation

2.3.3.1 Task Description

This technical case introduces a confidence-aware semantic scene completion algorithm [4] for visual semantic navigation, which uses a "confidence-aware semantic scene completion module" to explicitly model scene priors and uses this scene representation to guide the robot's navigation planning.

Given a local observation view of an indoor scene in the form of RGB-D images, the semantic scene completion module predicts the semantic information of the complete surrounding environment centered on the robot (a top-down bird's eye view). By learning the statistical data of multiple room layout structures, this module can use contextual clues to predict the complete room layout situation outside the robot's field of view in a typical indoor environment. Then, a learning-based navigation algorithm generates the next action based on the prediction of the complete 3D environment.

Since the scene prediction results inferred are used for the robot's next action planning, it is particularly important that the completion module can generate a confidence assessment of its prediction results to measure the accuracy of the

prediction results. For this purpose, the completion module will generate a dense self-supervised confidence map synchronously with its predicted completion map, using confidence estimation to guide the navigation strategy's trust level in the inference results of the completion module. This is very important for effectively guiding navigation and exploration.

The navigation module is trained using deep reinforcement learning [36], and its policy network takes the robot's current state and target object category as input and outputs the next action. The state is represented by the semantic completion map and confidence map, and the action is represented by a dense spatial action map [37]. In this action map, each pixel corresponds to an action, i.e., moving the robot to the spatial position corresponding to that pixel. Compared with sparse action representations (such as turn commands), this spatial action map representation method naturally aligns with the scene representation, and as shown in the ablation experiments, it can significantly improve performance [37–40].

This technical case focuses on applying semantic scene completion and confidence estimation to the object goal visual semantic navigation framework. By learning the contextual prior information of typical indoor environments, the algorithm can infer the scene representation outside the robot's local observation range, obtain a complete scene representation with confidence estimation, and thus guide the robot to efficiently explore unknown scenes and invisible areas, thereby providing more effective path planning strategies for downstream navigation tasks.



Chapter Knowledge Points

Deep Reinforcement Learning (DRL): Combining the perceptual ability of deep learning and the decision-making ability of reinforcement learning, it can directly control based on the input image, which is an artificial intelligence method closer to human thinking.

Visual Semantic Navigation: That is, in an unknown environment, generate actions for the active agent to navigate to the specified target object category. To complete this task, the algorithm should locate and navigate to an instance of this category at the same time. Compared with traditional point goal navigation, this task requires the agent to have a strong contextual prior of the indoor environment.

One-Hot Vector: Used in digital circuits and machine learning to represent a special bit tuple or vector, in which only one bit is allowed to be 1, and all other bits must be 0. It is called one-hot because there can only be one 1. If the situation is reversed, only one 0, the rest are 1, it is called one-cold. In statistics, dummy variables represent similar concepts.

2.3.3.2 Background Technology

- (1) **Scene Completion:** The SSCNav algorithm learns scene prior information through scene completion, which is an idea adopted by many recent works. Pathak et al. [41] proposed the ContextEncoder algorithm, which uses image completion to learn a representation method that can capture contextual information. Song et al. [30, 42] proposed to explicitly encode 3D structure and semantic information in the scene completion network. However, these works have not studied how to use these scene completion models in the context of motion planning, specifically, they have not provided a method to evaluate network prediction uncertainty, but this uncertainty is crucial for downstream motion planning algorithms. Jayaraman and Grauman [43–44] studied the use of panoramic image completion in the task of viewpoint selection. Ramakrishnan et al. [45] demonstrated the effectiveness of this learning strategy in some action perception applications. However, the image completion task is performed in the RGB color space, without modeling semantics or 3D structure. In addition, the actions used in these works are limited to camera rotation, so they cannot directly support navigation tasks that require translation actions.
- (2) **Point Goal Navigation:** The point goal navigation task has been a well-studied problem, with many existing works [46–49]. In the point goal navigation task, the goal position and egocentric observation are taken as inputs (images with color, depth, semantic segmentation, etc.), and the robot needs to choose and execute one of many possible actions until it reaches the unknown goal. However, semantic goal navigation, especially object goal navigation, is more challenging, because the robot does not know the location and appearance of the target object, so the system needs to estimate the target object’s location and the path to it at the same time.
- (3) **Semantic Goal Navigation:** Semantic goal navigation has received increasing attention in recent years, such as ObjectNav [50], whose goal is to navigate to a specified target object category, rather than a specific world position coordinate. For example, the self-adaptive visual navigation method (SAVN) [51] uses meta-learning to adapt to invisible environments and the goal-oriented semantic exploration. Exploration GOSE [52] constructed a global top-down view and used it to improve exploration efficiency. Learning value from videos (VLV) [53] learns statistical rules between object categories by watching YouTube videos. Another research method is to use semantic graphs to encode the scene structure for navigation [5, 54]. The target object is specified by text and the scene prior is learned through a language graph module, which mainly focuses on capturing the functional prior of objects with language. Unlike these works, this method uses a scene completion module to explicitly capture the scene prior. This scene completion module can naturally encode the functional prior knowledge and spatial relationship of object objects, which may not be reflected in text or 2D videos.

2.3.3.3 Technical Implementation

This method mainly consists of two parts:

- (1) Semantic Scene Completion Module: It accepts the robot's local observation results of the environment and infers the completed semantic representation $\mathcal{A}$ with confidence c_r .
- (2) Navigation Module: Accepts the estimated $\mathcal{A}$, c_r , and target object category to generate the next best action for the robot to move toward the target. As shown in Fig. 2.30, at each step in the navigation process, the robot will obtain a pair of RGB-D images (a), which are first used to predict the semantic segmentation of the robot's current view (b). Next, the past five steps of observation results are aggregated into a point cloud and projected as a top-down view semantic map (c). Then the scene completion network (i) completes the scene, and the completed top-down view map (d) and the observed semantic map (c) are used by the confidence network (j) to estimate the confidence of the predicted completion map, resulting in a completion confidence map (e). Again, the navigation network uses the top-down view completion map (d), the corresponding confidence map (e), and the target object category to predict the spatial action map (f), which is used to select the next action. Finally, the termination discriminator (h) uses the newly observed semantic segmentation map to determine whether the target has been reached.

The sample code summarizes the entire navigation process.



Sample Code

Input: RGB-D image observation, target object category

Output: Robot movement trajectory to reach the target object category

1. Initialization: (1) Uniformly sample a target object category g from the semantic category library; (2) Randomly generate a legal robot starting position;
2. REPEAT;
3. Perform semantic segmentation on the input RGB-D;
4. Generate point cloud pcd based on RGB-D re-projection;
5. Based on the point cloud and semantic segmentation results, project a map centered on the robot from a top-down view;
6. Perform semantic completion on this map;
7. Evaluate the semantic completion result to get a confidence map;
8. Input the target object category, confidence map, and completion map into the reinforcement learning network DQN to generate a spatial action map;
9. According to this action space map, choose the point with the highest value as the next action behavior and execute this action;
10. END.

Confidence-Aware Semantic Scene Completion

- (1) **Environment and Robot Settings:** The Habitat [50] and Matterport3D environments [55] are used to train and test this algorithm. The robot uses a first-person perspective RGB-D camera with a resolution of 480×640 , and the camera is 1.25 meters above the ground, looking down at a slight angle of 30° . With each step, the robot collects a new pair of RGB-D images and obtains semantic segmentation results using the existing work ACNet [56]. For the semantic segmentation network ACNet, 209,200 RGB-D images from 40 object categories in the Matterport3D environment are used to train it.
- (2) **Semantic Map from a Top-Down Perspective Centered on the Robot:** Fig. 2.31 shows the scene completion data process. By collecting depth maps and corresponding semantic observations centered on the robot (a), a 3D point cloud reconstruction results in the world coordinate system is obtained from the observed area (b) for point cloud projection from a top-down perspective to get an observation map centered on the robot. All observation data is used to generate a top-down target semantic map (c), and part of the observation images (1–4 views from all observation data) are used to generate the input semantic map for the completion network (d). To train the completion network, this method only calculates the loss in the observed area of the target map and ignores the unobserved area (e). As the robot moves in the environment, it integrates the past five steps of RGB-D images, semantic segmentation maps, and corresponding camera poses, where the camera poses are used to back-project and merge the past

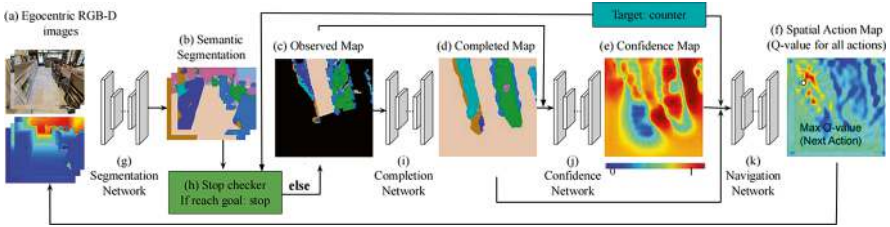


Fig. 2.30 Overview of SSCNav method

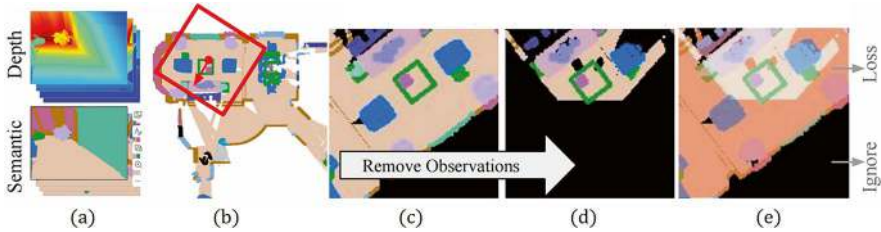


Fig. 2.31 Scene completion data. (a) Observation; (b) Reconstruction; (c) Target Map (all observations); (d) Completion Input (fewer observations); (e) Supervision

five steps of observation results into a 3D point cloud in a unified world coordinate system. (Note: Here, the ground truth pose data of the camera is provided by the environment, as camera pose estimation is not within the scope of this task) Then, based on the floor plane where the robot is located, points higher than 1.55 m or lower than the floor height are removed, and the remaining point cloud is projected from a top-down perspective at step t to get a top-down semantic map o_t centered on the robot. This semantic map o_t represents a $6\text{ m} \times 6\text{ m}$ local area centered on the robot and aligned with its direction (Fig. 2.31c). Each pixel in the top-down semantic map represents the semantic category of the object corresponding to the spatial position, represented by a one-hot vector with $(N + 1)$ channels, where N is the number of object categories, and the extra channel represents the unknown category. This robot-centered semantic map o_t is input into the scene completion network.

- (3) **Scene Completion Network:** The goal of the scene completion network is to output a completed top-down semantic completion map $\hat{o}_t$ based on the input robot-centered local observation semantic map o_t , where unobserved areas are filled with semantic information predicted by the scene completion network. The completion module is implemented by a fully convolutional neural network, which has one max pooling layer, four downsampling residual blocks, and five upsampling residual blocks [57]. The network is trained to minimize the pixel-wise cross-entropy loss between the predicted semantic label values and the true values on the “unobserved area.”
- (4) **Self-calibrating Confidence Estimation:** The semantic scene completion module predicts the layout of the environment around the robot based on the typical distribution of indoor environments learned from training data. However, in practice, the high variance and severe occlusion of scene layouts may result in high uncertainty in scene prediction results, which can mislead the navigation module.

To alleviate this problem, another branch is used to explicitly evaluate the prediction uncertainty of the completion network, which outputs the corresponding confidence map $c_t \in [0, 1]^{N \times M}$. The confidence map branch is trained in a self-calibrating manner by comparing the network prediction results with the ground truth, obtaining the target accuracy map $c_{tg} \in [0, 1]^{N \times M}$. Then, the task of the confidence network is to predict the semantic correctness value of each pixel based on the input observation o_t and the completed semantic map $\hat{o}_t$. The confidence network is trained only in the “unobserved area” by minimizing the pixel-level MSE loss between c_t and c_{tg} .

- (5) **Training:** To obtain the training data for the scene completion network, this method allows the robot to randomly explore the environment, aggregate its observation data and estimated data (such as RGB and depth map), and reconstruct the 3D scene. During the exploration process, the scene completion network is trained by randomly deleting observed view data, and the task of the network is to infer the missing data. This training step will ignore unexplored areas in the environment (as shown in Fig. 2.31). This learning scheme allows

the system to learn from any new scene through navigation without having to completely reconstruct the entire scene, thus enabling scene exploration based on visual semantic navigation. In this method, a total of 52,300 top-down ground truth maps were generated for training, each ground truth map paired with 4 different input observations, and views were randomly deleted.

Visual Semantic Navigation Based on Reinforcement Learning

- (1) DQN formula: The navigation task is formalized as a Markov decision process. Given the target object category g and the current state s_t , the navigation policy $\pi(s_t)$ outputs the next action a_t of the robot, then receives the next state s_{t+1} and the reward value r_t from the environment. The learning goal is to find a series of actions that maximize the total expected future return $\sum_{i=t}^{\infty} \gamma^{i-t} r_i$, where $\gamma \in (0,1)$ is the discount sum of all future returns.

This method uses offline Q-learning [58], which learns a policy $\pi(s_t)$ that selects the action that maximizes the Q function $\arg \max_{a_t} Q_{\theta}(s_t, a_t)$ (i.e., the state-action value function), which is a fully convolutional neural network [57, 59] parameterized by θ . This method uses the Double-DQN learning target [60] to train the robot. That is, at each step t , the goal is to minimize: $\mathcal{L} = |r_t + \gamma Q_{\theta^-}(s_{t+1}, \arg \max_{a_{t+1}} Q_{\theta}(s_{t+1}, a_{t+1})) - Q_{\theta}(s_t, a_t)|$, where (s_t, a_t, r_t, s_{t+1}) is a set of data uniformly sampled from the robot's replay buffer. The parameters of the target network θ^- remain fixed between updates and are updated at a lower frequency.

- (2) State representation: The state representation s_t includes three parts: (1) the top-down semantic completion map $\mathcal{O}\tilde{\mathcal{F}}$; (2) the corresponding confidence estimate c_t ; and (3) the target object category g . The target object category is encoded as a one-hot vector with $(N + 1)$ channels (N object categories plus background). This vector is tiled to the same shape as c_t and $\mathcal{O}\tilde{\mathcal{F}}$. The three elements are concatenated along the channel as the state representation.
- (3) Action representation: The action space A is represented as a spatial action map similar to Wu et al. [37]. The spatial action map has the same spatial size as the state representation, where each pixel in the top-down overhead view corresponds to the endpoint of local navigation in the map centered on the robot. At each step, after the robot selects a pixel from the spatial action map, it turns to the position in the environment corresponding to that point and moves forward one step, with a maximum step length of 0.25 m (collisions may occur). To select the best action, the navigation network estimates the expected Q values of all possible actions, represented as a heat map (see Fig. 2.32). During testing, actions are selected by greedily choosing the pixel with the maximum Q value.

This Q -value represents alignment in space and anchoring to the scene representation, thus enabling faster learning of complex behaviors in navigation. In addition,

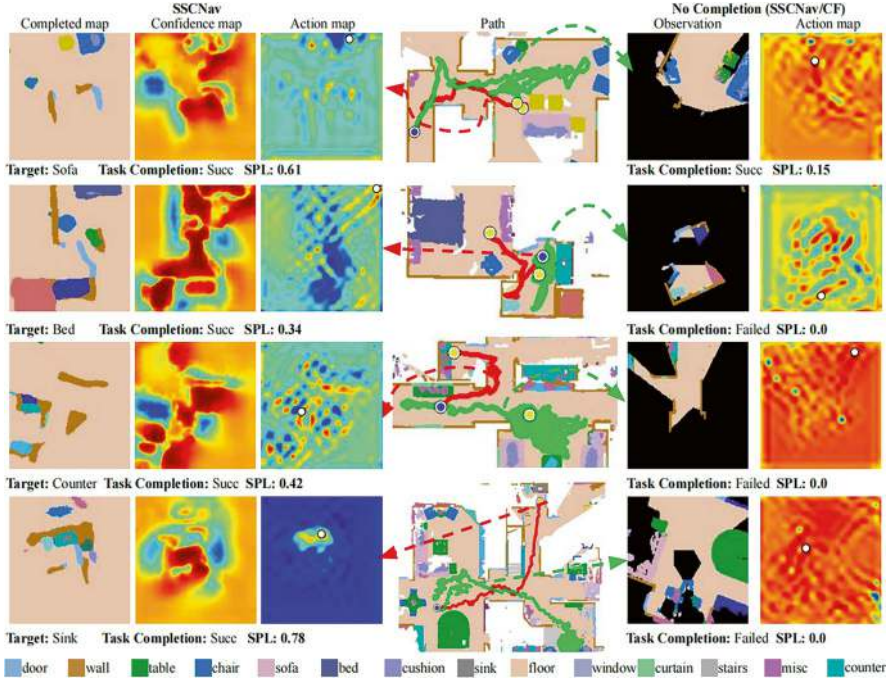


Fig. 2.32 Qualitative results of semantic navigation

this spatial representation allows the visualization of the Q -value of each local navigation endpoint and the behavior of the navigation strategy on a heatmap.

- (4) Reward: At time step t , the robot receives a reward r_t , which is the sum of the following items:
 - (1) A life penalty of 0.01 to encourage shorter paths.
 - (2) If the step distance is < 0.125 m, the penalty is -0.25 , to encourage movement and penalize collisions.
 - (3) Incremental distance reward is proportional to the change in distance from the agent to the nearest target object.
 - (4) If the agent is successful, it receives a success reward of $+10$.
- (5) Navigation termination discriminator: As part of the task, the system needs to determine whether it has successfully reached the target. To simulate this ability, a termination discriminator (Fig. 2.30h) was designed, which uses semantic and depth information to check whether the target object category has been reached. From the robot's current location, its camera is rotated four times along the z -axis to obtain paired observation results. If in at least one pair of observations, there are enough pixels (> 5000) within 1 m of the robot that belong to the target object category, the termination discriminator returns true and terminates the navigation. If the robot has moved 500 steps (maximum number), the navigation will also terminate.

- (6) Task completion: With each move of the robot, the environment checks whether the robot is successful and provides rewards accordingly. According to the literature [61], the following success indicators are used (please note that validity is negligible in this method setting):
- (1) Intentionality: The robot’s termination discriminator has returned true;
 - (2) Proximity: The robot is less than 1 m away from a target instance;
 - (3) Visibility: At least one target instance is visible.
- (7) Training: Each training/testing set T contains four elements: Starting position p , starting direction f , scene h , and target object category g . For the training set T_k , the target object category is uniformly sampled, then a training scene is selected from the Matterport3D training set, and a legal starting position p is randomly selected. If the starting position: (1) is passable; (2) is located in one of the valid room types (bathroom, bedroom, dining room, kitchen, living room, laundry room, and family activity room) to avoid starting from outdoors; (3) is far from all target objects (the Euclidean distance and geodesic distance to any target instance are both greater than $r + D_{succ}$, where r is the radius of the object instance, D_{succ} is the success distance (1 m)), then this starting point is considered legal.

If no suitable starting point is found after 100 attempts, the target object category and scene are searched again. Otherwise, the robot will face a random direction and prepare to start navigation.

The entire semantic navigation qualitative results are shown in Fig. 2.32. In the visualization path (column 4), the red path shows the robot’s navigation path using the semantic scene completion module (SSCNav), the green path is the robot’s path without using the semantic scene completion module (SSCNav-CF) (CF: without scene completion and confidence, indicating no scene completion module), the blue dot represents the starting position, and the yellow dot represents the ending position. Please note that in the first row, the robots of both methods successfully reached their targets (sofa), but SSCNav-CF made a huge mistake, making it difficult to find the target.

2.3.3.4 Results Presentation

In this section, experimental results are provided to verify the effectiveness of the method, using the standard training and testing split ratio of Matterport3D [55] to run all experiments on the Habitat platform [50]. In all experiments, the robot was tested in new rooms.

Semantic Scene Completion

- (1) Metrics: Evaluating the performance of scene completion. This algorithm is tested using 715 invisible house samples from Matterport3D, using category-level pixel intersection over union (IoU) as the evaluation metric. IoU is calcu-

lated in areas that are not observed in the input top-down view but are observed in the ground truth [42].

(2) Results: Fig. 2.33 shows the quantitative results of the scene completion network using the estimated semantic segmentation (completion-seg) and ground truth semantic segmentation (completion-gt) as input observations. The figure shows the intersection over union (IoU) of the estimated semantic labels and ground truth labels in the unobserved area when using ground truth segmentation as input (blue) and estimated segmentation as input (red). Six categories with an IoU of zero (beams, blinds, gym equipment, boards, ceilings, and miscellaneous) are excluded. Figure 2.34 shows the qualitative results and the esti-

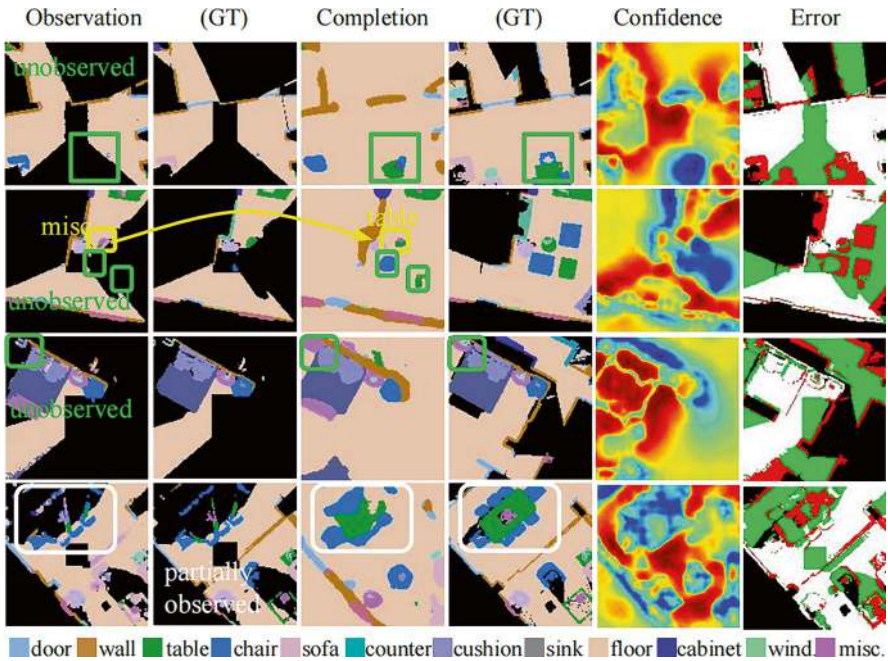


Fig. 2.33 Scene completion results

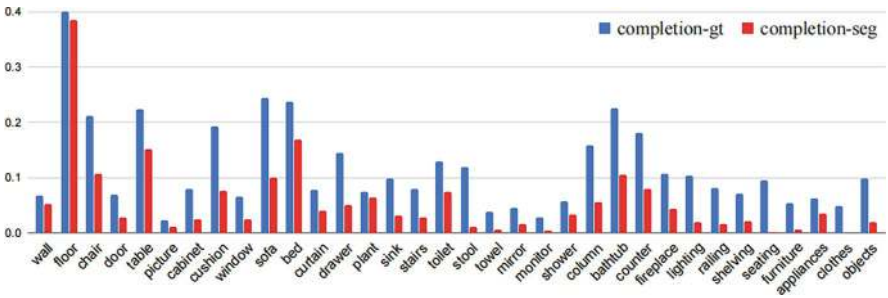


Fig. 2.34 Semantic scene completion results

mated confidence map. From left to right are the input observed segmentation prediction and ground truth segmentation, scene completion prediction results and ground truth, confidence estimation, and error map (white—observed area, red—incorrect completion, and green—correct completion). The results show that the scene completion network can:

- (1) Complete the geometric shapes of partially observed objects—Fig. 2.34, row 4.
- (2) Infer unobserved objects based on context information—Fig. 2.34, row 3 shows an example where the algorithm is able to infer the existence of an unobserved bedside table based on the observed bedside table and infer the unobserved table and chair in rows 1 and 2.
- (3) Correct the erroneous segmentation results of the semantic segmentation network in the input observation—Fig. 2.34, row 2 shows an example where the completion network corrects the erroneous prediction of miscellaneous (misc) to table (table). At the same time, the completion model can also produce erroneous prediction results (marked in red in Fig. 2.34). Generally, the confidence scores generated by the network reflect the credibility of the prediction results (lower scores indicate less credibility).

Semantic Navigation Results

- (1) Metrics: Use the test set provided by the Habitat 2020 ObjectNav Challenge [61] to evaluate this navigation model. To evaluate the performance of visual semantic navigation, the standard navigation success rate and SPL (Success weighted by Path Length) are used as metrics [62] during testing. Please note that there may be multiple object instances belonging to the same target category. In this case, when starting to calculate the shortest path, the nearest target object instance is directly selected.
- (2) Comparison with previous work: Here, the SAVN method of Wortsman et al. [51] is evaluated and compared. For comparison, the SAVN method is fine-tuned and tested using the same settings as this method, and summarized in Table 2.2, where /SA represents using sparse actions, CF represents no scene completion and confidence estimation, F represents no confidence estimation, /BC represents visibility-based binary confidence, G+ represents using ground truth segmentation for the observed scene. Because the SAVN method initially only had real-time penalties and success rewards during training, the gap is narrowed by adding additional continuous distance rewards (the same as this method). The results show that the method used in this technical case is significantly superior to the SAVN method in all object categories. The reasons may be twofold: Firstly, the SAVN method uses a sparse action representation, i.e., $A = \{\text{MoveAhead}, \text{RotateLeft}, \dots, \text{Done}\}$, which is harder to learn from the mapping of observed to action classes compared to the dense predictions generated by the spatial action graph; Secondly, the SAVN method does not explicitly

Table 2.2 Navigation results (success rate/SPL)

Model	Bed	Counter	Shower	Sink	Sofa	Table	Toilet	Avg
G + SSCNav-CF (GOSE [19])	0.083/0.024	0.136/0.062	0.200/ 0.101	0.207/0.098	0.377/0.182	0.591/0.380	0.036/0.003	0.387/0.232
G + SSCNav-F	0.083/0.061	0.121/0.054	0.043/0.01	0.268/ 0.158	0.208/0.129	0.574/0.394	0.071/0.004	0.357/0.235
G + SSCNav	0.104/0.057	0.227/0.092	0.257/0.073	0.280/0.149	0.245/0.115	0.656/0.425	0.143/0.061	0.438/0.259
SAVN [18]	0.000/0.000	0.000/0.000	0.000/0.000	0.000/0.000	0.000/0.000	0.018/0.018	0.000/0.000	0.009/0.009
SSCNav/SA	0.000/0.000	0.015/0.002	0.000/0.000	0.049/0.019	0.000/0.000	0.206/0.086	0.000/0.000	0.109/0.045
SSCNav-CF (GOSE [19])	0.021/0.002	0.015/0.005	0.043/0.009	0.037/0.027	0.151/0.055	0.394/0.219	0.000/0.000	0.218/0.118
SSCNav-F	0.042/0.035	0.076/0.034	0.000/0.000	0.122/0.028	0.019/0.009	0.385/0.216	0.000/0.000	0.217/0.117
SSCNav/BC	0.000/0.000	0.121/0.027	0.057/0.006	0.232/0.083	0.038/0.009	0.409/0.275	0.036/0.010	0.252/0.150
SSCNav	0.042/0.040	0.152/0.040	0.200/0.059	0.268/0.097	0.057/0.029	0.388/0.261	0.107/0.037	0.271/0.157

model semantic object classes in the scene representation, which may make learning more difficult.

Another work is the GOSE method [52], which is the winner of the Habitat 2020 ObjectNav Challenge [61]. The GOSE method uses a similar top-down semantic map and spatial action representation. However, due to the code not being public, the main difference is the lack of a semantic scene completion module, which can be referred to the performance of [SSCNav without(−) scene(C) ompletion and con(F) idence] in Table 2.2 as an attempt to reproduce the GOSE method.

- (3) Is confidence-aware scene completion helpful? To test the effect of using the semantic scene completion module in navigation, the experiment compared the use and nonuse of the semantic scene completion module ([SSCNav-CF] compared to [SSCNav-F] compared to [SSCNav], C, scene(C)ompletion F, con(F) idence). By comparing [SSCNav-CF] and [SSCNav-F], it can be observed that directly using scene completion does not affect the navigation success rate, which may be because the errors and noise in the scene completion results still mislead the navigation planning, so the output of scene completion (after softmax) The original confidence information carried is not enough.

When confidence estimation is added, this problem is largely alleviated. [SSCNav] surpassed [SSCNav-CF] with a success rate of +5.3% and a SPL of +3.9%. Even with completion errors, [SSCNav] guided by confidence awareness can recover and surpass [SSCNav-CF] in most categories. For some categories where [SSCNav] is lower than [SSCNav-CF], the SPL value is much higher (e.g., table −0.6% success rate, +4.2% SPL). The results show that the confidence-aware semantic scene completion module is particularly useful for objects with strong context bias (e.g., toilets only appear in bathrooms), but for more common objects, such as sofas and tables that can appear in almost any room, context bias is not very useful.

- (4) What has confidence estimation learned? Since confidence estimation is crucial for the model to benefit from scene completion, this method wants to verify whether confidence estimation is better than simply setting the visible area as “trustworthy” and the invisible area as “untrustworthy”? To answer this question, the experiment included an additional baseline [SSCNav/BC] (BC, binary confidence), where confidence is modeled according to scene visibility: the visible area is valued at 1, and the invisible area is valued at 0. The results show that [SSCNav] has a + 1.9% success rate and + 0.7% SPL compared to [SSCNav/BC]. This result verifies that although scene visibility is highly correlated with network uncertainty, confidence estimation still carries more useful information.
- (5) The effect of spatial action representation: To test the benefits of using spatial action maps as action representations, the experiment compared [SSCNav] with [SSCNav/SA] (/SA, sparse action). [SSCNav/SA] replaces the output map of the navigation model with a sparse action probability set, which is a common strategy for such navigation methods. The experiment uses the following action space: {Turn right k degrees, then moveforward, where $k \in$

[0,45,90,135,180,225,270,315]]. At each step, the robot chooses the action with the highest probability rather than the pixel with the highest Q value. This method not only makes the navigation model lose its interpretability but also sees a significant drop in success rate (−16.2%) and SPL (−14.8%). This result is consistent with previous work [38].

- (6) Performance prediction: Here, the navigation performance when the ground truth semantic segmentation of the observation area is available was evaluated. The three experimental results related to [G + SSCNav] in Table 2.2 show the navigation performance using the ground truth segmentation of the observation area as input. Although the input of the ground truth significantly improves the algorithm performance, the results still have a similar trend—even with accurate semantic segmentation as input, semantic scene completion still improves navigation performance. This result indicates that as the general semantic segmentation algorithm progresses, the performance of this system will also improve.

2.3.3.5 Summary

In this technical case, the introduction of SSCNav into the visual semantic navigation task is introduced. The algorithm uses a confidence-aware semantic scene completion module to better allow the robot to understand the surrounding environment, predict the typical environment distribution in the invisible area of the robot's environment by learning the contextual prior information of typical indoor environments, and use this scene completion result to promote its exploration of unknown environments, making the navigation module perform more reasonable action planning. Finally, the experiment shows that this exploration method based on scene completion, which learns the contextual prior information of typical indoor environments, can improve the performance of visual semantic navigation in terms of success rate and efficiency.

2.4 Summary

This chapter introduces two methods for autonomous exploration of unknown scenes in two tasks, namely, scene exploration in autonomous scene reconstruction and scene exploration in visual semantic navigation. For tasks involving mobile robots, scene exploration is the basis and key to the execution of tasks by mobile robots. However, due to the occlusion problem caused by the complex layout of the scene and the problem of limited camera view, the robot's observation of the environment usually only contains a local area of the entire environment, making it difficult for the robot to obtain global scene information, and can only use local observation data for behavior planning, making it difficult for the robot's behavior to achieve global optimization.

In response to this type of problem, this chapter proposes a strategy for guiding robot movement using the rich contextual prior knowledge in the scene. For autonomous scene reconstruction tasks, an object-aware global exploration strategy and a local scanning strategy for objects guided by 3D shape priors are introduced. The robot scans the objects in the scene one by one, just like humans do, to achieve global exploration of the scene. Meanwhile, for each object scan, similar models from its 3D model database are used as priors to guide the robot to find the best local scanning viewpoint, complete the object scan, and iterate this process to achieve autonomous scene scanning and reconstruction tasks. For visual semantic navigation tasks, a navigation strategy based on a confidence-aware semantic scene completion algorithm is introduced, allowing the robot to use deep reinforcement learning to train in multiple indoor scenes and learn the typical distribution of the environment as the contextual prior knowledge of the indoor scene. When the robot enters an unknown scene, it can reasonably predict the complete room layout outside its field of view based on its local observation information and prior knowledge, and make a confidence estimate of this prediction, using this as a reference to plan the robot's navigation path. Both methods have been proven feasible and superior through relevant technical cases.

2.5 Discussion

- (1) Please list some applications of scene exploration and try to compare the differences between these methods.
- (2) Please briefly describe the visual semantic navigation task and the point goal navigation task, and analyze their differences and connections.
- (3) Please briefly describe the role of semantic segmentation in scene exploration in autonomous scene reconstruction tasks.
- (4) Please briefly analyze the role of semantic scene completion in visual semantic navigation.
- (5) Practice: Please try to build a virtual scene in the Gazebo platform, implement autonomous scene exploration and reconstruction based on a greedy strategy, and be able to reconstruct a 3D scene (voxel representation) based on the collected RGB-D data, making each robot movement choose an action that maximizes the information gain of the 3D reconstruction model.
- (6) Practice: Please try to build a DQN-based visual semantic navigation network on the Habitat platform, so that the robot can autonomously decide the next action based on the currently observed RGB-D data. If the target semantic category is found within a certain number of steps, it is considered a navigation success, otherwise, it is considered a navigation failure. Please train and evaluate on the Matterport3D dataset.

References

1. XU K, ZHENG L T, YAN Z H, et al. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields[J]. *Acm Transactions on Graphics*, 2017, 36(6): 1–15.
2. LIU L, XIA X, SUN H, et al. Object-aware guidance for autonomous scene reconstruction[J]. *Acm Transactions on Graphics*, 2018, 37(4): 1–12.
3. ZHENG L, ZHU C, ZHANG J, et al. Active scene understanding via online semantic reconstruction[J]. *Computer Graphics Forum*, 2019, 38(7): 103–114.
4. LIANG Y, CHEN B, SONG S. Sscnav: confidence-aware semantic scene completion for visual semantic navigation[C]//*IEEE international conference on robotics and automation*. New York: IEEE, 2021: 13194–13200.
5. WEI Y, XIAOLONG W, FARHADI A, et al. Visual semantic navigation using scene priors [arxiv\[EB/OL\]](https://arxiv.org/abs/1810.06543) (2018) [http://arxiv.org/abs/1810.06543](https://arxiv.org/abs/1810.06543).
6. SINGH CHAPLOT D, SALAKHUTDINOV R, GUPTA A, et al. Neural topological slam for visual navigation[C]//*IEEE Conference on Computer Vision and Pattern Recognition*. NEW YORK: IEEE, 2020: 12875–12884.
7. CHARROW B, KAHN G, PATIL S, et al. Information-theoretic planning with trajectory optimization for dense 3D mapping[C]//*11th conference on robotics-science and systems*. Cambridge: Mit Press, 2015.
8. KRAININ M, CURLESS B, FOX D. Autonomous generation of complete 3D object models using next best view manipulation planning[C]//*IEEE international conference on robotics and automation*. New York: IEEE, 2011: 5031–5037.
9. WU S, SUN W, LONG P, et al. Quality-driven poisson-guided autoscanning[J]. *Acm Transactions on Graphics*, 2014, 33(6): 1–12.
10. KRIEGEL S, RINK C, BODENMULLER T, et al. Next-best-scan planning for autonomous 3D modeling[C]//*25th IEEE/RSJ international conference on intelligent robots and systems*. New York: IEEE, 2012: 2850–2856.
11. ENGELHARD N, ENDRES F, HESS J, et al. Real-time 3D visual slam with a hand-held rgb-d camera[C]//*Proc of the RGB-D workshop on 3D perception in robotics at the european robotics forum*, Vasteras: ERF, 2011: 1–15.
12. SALAS-MORENO R F, NEWCOMBE R A, STRASDAT H, et al. Slam plus plus: simultaneous localisation and mapping at the level of objects[C]//*26th IEEE conference on computer vision and pattern recognition*. New York: IEEE, 2013: 1352–1359.
13. WHELAN T, LEUTENEGGER S, SALAS-MORENO R E, et al. ElasticFusion: dense slam without a pose graph[C]//*11th conference on robotics-science and systems*. Cambridge: Mit Press, 2015.
14. XU K, SHI Y, ZHENG L T, et al. 3D attention-driven depth acquisition for object identification[J]. *Acm Transactions on Graphics*, 2016, 35(6): 1–14.
15. FISHER M, RITCHIE D, SAVVA M, et al. Example-based synthesis of 3D object arrangements[J]. *Acm Transactions on Graphics*, 2012, 31(6): 1–11.
16. VALENTIN J, VINEET V, CHENG M M, et al. Semanticpaint: interactive 3D labeling and learning at your fingertips[J]. *Acm Transactions on Graphics*, 2015, 34(5): 1–17.
17. NAN L, XIE K, SHARF A. A search-classify approach for cluttered indoor scene understanding[J]. *Acm Transactions on Graphics*, 2012, 31(6): 1–10.
18. ZHANG Y, XU W, TONG Y, et al. Online structure analysis for real-time indoor scene reconstruction[J]. *Acm Transactions on Graphics*, 2015, 34(5): 1–13.
19. XU K, HUANG H, SHI Y, et al. Autoscanning for coupled scene reconstruction and proactive object analysis[J]. *Acm Transactions on Graphics*, 2015, 34(6): 1–14.
20. THRUN S. Robotic mapping: A survey[J]. 2002: 1–35.
21. AGARWAL S, SNAVELY N, SEITZ S M, et al. Bundle adjustment in the large[C]//*11th european conference on computer vision*. Berlin: Springer-Verlag Berlin, 2010: 29–42.
22. CHOI S, ZHOU Q, KOLTUN V. Robust reconstruction of indoor scenes[C]//*IEEE conference on computer vision and pattern recognition*. New York: IEEE, 2015: 5556–5565.

23. NEWCOMBE R A, IZADI S, HILLIGES O, et al. KinectFusion: real-time dense surface mapping and tracking[C]//10th IEEE/ACM international symposium on mixed and augmented reality. New York: IEEE, 2011: 127–136.
24. RAMANAGOPAL M S, LE N J. Motion planning strategies for autonomously mapping 3D structures arxiv [EB/OL] (2016) <http://arxiv.org/abs/1602.06667>.
25. ZENG A, SONG S, NIESSNER M, et al. 3DMatch: learning local geometric descriptors from rgb-d reconstructions [C] //30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 199–208.
26. FAN X, ZHANG L, BROWN B, et al. Automated view and path planning for scalable multi-object 3D scanning [J]. *Acm Transactions on Graphics*, 2016, 35(6): 1–13.
27. LOW K L, LASTRA A. An adaptive hierarchical next-best-view algorithm for 3D reconstruction of indoor scenes [C] //Proceedings of 14th pacific conference on computer graphics and applications (Pacific Graphics 2006). Princeton, N J: Citeseer, 2006: 1–8.
28. TATENO K, TOMBARI F, NAVAB N. Real-time and scalable incremental segmentation on dense slam [C] //IEEE/RSJ international conference on intelligent robots and systems. New York: IEEE, 2015: 4465–4472.
29. LLOYD S P. Least-squares quantization in pcm [J]. *IEEE Transactions on Information Theory*, 1982, 28(2): 129–137.
30. SONG S, YU F, ZENG A, et al. Semantic scene completion from a single depth image [C] //30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 190–198.
31. DAI A, CHANG A X, SAVVA M, et al. ScanNet: richly-annotated 3D reconstructions of indoor scenes [C] //30th IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2017: 2432–2443.
32. GAZEBO. The gazebo project. [EB/OL] (2013) <http://wiki.ros.org/gazebo>.
33. PALACIOS J, ZHANG E. Rotational symmetry field design on surfaces [J]. *Acm Transactions on Graphics*, 2007, 26(3): 1–10.
34. HORNUNG A, WURM K M, BENNEWITZ M, et al. Octomap: an efficient probabilistic 3D mapping framework based on octrees [J]. *Autonomous Robots*, 2013, 34(3): 189–206.
35. NIESER M, PALACIOS J, POLTHIER K, et al. Hexagonal global parameterization of arbitrary surfaces [J]. *IEEE Transactions on Visualization and Computer Graphics*, 2012, 18(6): 865–878.
36. QI S, ZHU Y, HUANG S, et al. Human-centric indoor scene synthesis using stochastic grammar [C] //31st IEEE/CVF conference on computer vision and pattern recognition. New York: IEEE, 2018: 5899–5908.
37. WU J, SUN X, ZENG A, et al. Spatial action maps for mobile manipulation [C] //16th conference on robotics-science and systems. Cambridge: Mit Press, 2020.
38. ZENG A, SONG S, YU K, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching [J]. *International Journal of Robotics Research*, 2022, 41(7): 690–705.
39. ZENG A, SONG S R, WELKER S, et al. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning [C] //25th IEEE/RSJ international conference on intelligent robots and systems. New York: IEEE, 2018: 4238–4245.
40. ZENG A. Learning visual affordances for robotic manipulation [D]; Princeton University, 2019.
41. PATHAK D, KRAHENBUHL P, DONAHUE J, et al. Context encoders: Feature Learning by Inpainting [C] //2016 IEEE Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2016: 2536–2544.
42. SONG S, ZENG A, CHANG A, et al. Im2pano3D: Extrapolating 360 Degrees Structure and Semantics Beyond the Field of View [C] //31st IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2018: 3847–3856.
43. JAYARAMAN D, GRAUMAN K. Look-ahead Before You Leap: End-to-end Active Recognition by Forecasting the Effect of Motion [C] //14th European Conference on Computer Vision. CHAM: Springer International Publishing Ag, 2016: 489–505.

44. JAYARAMAN D, GRAUMAN K. Learning to Look Around: Intelligently Exploring Unseen Environments for Unknown Tasks [C] //31st IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2018: 1238–1247.
45. RAMAKRISHNAN S K, JAYARAMAN D, GRAUMAN K. Emergence of Exploratory Look-around Behaviors Through Active Observation Completion [J]. *Science Robotics*, 2019, 4(30): 1–12.
46. SAVVA M, CHANG A X, DOSOVITSKIY A, et al. Minos: Multimodal Indoor Simulator for Navigation in Complex Environments [EB/OL] (2017) <https://arxiv.org/abs/1712.03931>.
47. XIA F, ZAMIR A R, HE Z Y, et al. Gibson Env: Real-world Perception for Embodied Agents [C] //31st IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2018: 9068–9079.
48. YAN C, MISRA D, BENNETT A, et al. Chalet: Cornell House Agent Learning Environment [EB/OL] (2018) <https://arxiv.org/abs/1801.07357>.
49. WIJMANS E, KADIAN A, MORCOS A, et al. DD-PPO: Learning Near-perfect Pointgoal Navigators from 2.5 Billion Frames [EB/OL] (2019) <https://arxiv.org/abs/1911.00357>.
50. SAVVA M, KADIAN A, MAKSYMETS O, et al. Habitat: A Platform for Embodied AI Research [C] //IEEE/CVF International Conference on Computer Vision. New York: IEEE, 2019: 9338–9346.
51. WORTSMAN M, EHSANI K, RASTEGARI M, et al. Learning to Learn How to Learn: Self-adaptive Visual Navigation Using Meta-learning [C] //IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2019: 3743–6752.
52. CHAPLOT D S, GANDHI D, GUPTA A, et al. Object Goal Navigation Using Goal-oriented Semantic Exploration Arxiv [EB/OL] (2020) <http://arxiv.org/abs/2007.00643>.
53. CHANG M, GUPTA A, GUPTA S. Semantic Visual Navigation by Watching YouTube Videos [J]. *Advances in Neural Information Processing Systems*, 2020, 33: 4283–4294.
54. QIU Y, PAL A, CHRISTENSEN H I. Target Driven Visual Navigation Exploiting Object Relationships [EB/OL] (2020) <https://arxiv.org/abs/2003.06749>.
55. CHANG A, DAI A, FUNKHOUSER T, et al. Matterport3D: Learning from RGB-D Data in Indoor Environments [C] //International Conference on 3D Vision. New York: IEEE, 2017: 667–676.
56. HU X, YANG K L, FEI L, et al. ACNet: Attention Based Network to Exploit Complementary Features for RGBD Semantic Segmentation [C] //26th IEEE International Conference on Image Processing. New York: IEEE, 2019: 1440–1444.
57. HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition [C] //2016 IEEE Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2016: 770–778.
58. MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. *Nature*, 2015, 518(7540): 529–533.
59. SHELHAMER E, LONG J, DARRELL T. Fully convolutional networks for semantic segmentation[J]. *IEEE transactions on pattern analysis and machine intelligence*, 2017, 39(4): 640–651.
60. VAN HASSELT H, GUEZ A, SILVER D, et al. Deep reinforcement learning with double q-learning[C]//30th Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence. PALO ALTO: Assoc Advancement Artificial Intelligence, 2016: 2094–2100.
61. BATRA D, GOKASLAN A, KEMBHAVI A, et al. Objectnav revisited: on evaluation of embodied agents navigating to objects[EB/OL] (2020) <https://arxiv.org/abs/2006.13171>.
62. ANDERSON P, CHANG A, CHAPLOT D S, et al. On evaluation of embodied navigation agents[EB/OL] (2018) <https://arxiv.org/abs/1807.06757>.

Chapter 3

Scene Structure Understanding



3.1 Overview

Scene understanding has always been a focus in computer vision. When a robot enters an unfamiliar scene, its primary task is to understand and recognize the real environment, which requires the robot to have the ability to parse visual scenes and effectively perceive, understand, and interpret the content in the scene.

Scene understanding is the process of perceiving, analyzing, and interpreting observed dynamic scenes through sensor networks, which are usually required to be real time. This process mainly matches the signal information from the sensors perceiving the environment with the models humans use to understand the scene and adds and extracts semantics to the sensor data describing the scene. This scene often contains different types of physical objects, which all have more or less interaction with each other and the environment.

Scene understanding is influenced by cognitive vision, including detection, localization, recognition, and understanding, which are four levels of general computer vision functions. However, a scene understanding system is not just about detecting visual features such as corners, edges, and moving areas but also needs to extract information related to the physical world, which is often more meaningful for the human world. The task of scene understanding also requires giving robots' cognitive abilities to achieve more robust, resilient, and adaptive computer vision functions, to achieve the ability to learn, adapt, and balance alternative solutions, and then to achieve the ability of advanced deduction and reasoning strategies.

The key to a scene understanding system is that it can show robust performance even in scenes that it did not foresee when it was designed. Therefore, a scene understanding system should be able to predict events and adjust its operations accordingly. Ideally, a scene understanding system should be able to adapt to the current environment, predict the state of the future environment by predicting the user's potential intentions or behaviors through the context information of the environment, and interact with the entities in the scene.

The task of 3D scene understanding is the process to perceive, understand, and interpret the content in the scene from 3D data, focusing on both geometric and semantic information, including recognizing and locating all objects in the scene, as well as the context information of the objects and the relationship between the objects. This thorough understanding is of great significance for various applications such as robot navigation, augmented reality, and virtual reality. The current work on 3D scene understanding includes perception tasks, such as instance segmentation, semantic segmentation, 3D objects detection and classification, and scene graph generation tasks. Although these efforts attempt to mine different levels of scene semantic information, their methods are essentially to use the contextual information of the scene and the relationships between objects to improve the accuracy of each object's prediction.

In the following chapter, we will explain the basic principles and research status of each scenario understanding method at home and abroad, and analyze its advantages, disadvantages, and applicable scenarios from the two levels of understanding of unstructured scene knowledge and structured scene knowledge (Sect. 3.2). Subsequent sections (Sects. 3.3 and 3.4) will introduce some practices related to 3D scene understanding (including point cloud semantic segmentation and object classification, and the task of generating scene graphs for 3D scenes), and finally summarize the content of this chapter (Sect. 3.5).

This chapter uses the open-source code of the PointNet network as the basic learning content for the task of indoor point cloud scene semantic segmentation and classification, and the open-source code of the 3DSSG (3D semantic scene graph) network as the basic material for the task of generating indoor scene point cloud scene graphs. For related resource acquisition, the following SN-branded short-URLs can be scanned by the SN More Media app.

► sn.pub/8434zv

3.2 Related Work

3.2.1 *Indoor Scene Understanding-Based Object Detection and Classification*

Current point cloud processing technology already supports direct point-by-point operations on 3D objects or scenes [1]. To ensure that deep learning technology can migrate from 2D vision to enhance 3D object-oriented recognition performance, 3D scene understanding tasks are usually object-centered recognition or detection tasks, including semantic scenes, semantic segmentation [2, 3], scene instance segmentation [4–6], scene object detection [7, 8], and classification tasks [6]. Among them, point cloud semantic segmentation lets the machine understand the scene point cloud collected by the sensor and predict the semantic category to which each point in the point cloud belongs. 3D object classification is to let the machine

process, analyze, and understand 3D objects to identify the type of 3D objects. These works mainly focus on unstructured scene knowledge, such as object semantic information, and usually use environmental context information to improve the accuracy of understanding object classes in the scene. Because point cloud semantic segmentation and 3D object classification algorithms greatly help machines understand the world, they have great research value and wide application potential, mainly including the following aspects:

- (1) Autonomous driving. Autonomous vehicles first need to use sensor equipment such as lidar to perceive and detect the surrounding environment of the car, and to automatically understand the surveyed data to distinguish pedestrians, vehicles, and traffic signs. Then in the behavior decision stage, algorithms are called based on the results of the previous step to make a series of decisions and perform autonomous driving operations. It can be seen that the perception and understanding of the environment is the premise and basis of autonomous driving technology. 3D vision has advantages such as higher accuracy compared to 2D vision, so in the field of autonomous driving, 3D point cloud understanding technology is playing an increasingly important role.
- (2) 3D face recognition. 3D face recognition is to collect the 3D information of the face in detail and ensure that the key information of the face can be completely entered. Based on this, the algorithm compares with the existing 3D face information in the database to achieve identity authentication. Compared with image face recognition, 3D face recognition is more secure, so in the context of the rapid development of payment technology, 3D face payment will gradually replace 2D face payment. For example, Alibaba Group is vigorously promoting its new retail business. At present, Alibaba has jointly launched a device for 3D face payment with Orbbec. With Alibaba's strong market driving force, 3D face payment devices have been applied in many physical stores.
- (3) Robot technology. Intelligent robots should have three aspects of capabilities: Perceiving the environment, interacting with real objects, and effectively combining perception and interaction. Currently, in order to improve the level of robot intelligence and enhance the robot's autonomous action capabilities, a basic issue is how to effectively implement the robot's accurate analysis and understanding of the 3D environment in which it is located, so as to provide it with precise location information. Scene point cloud understanding, as a basic problem of computer 3D vision, can lay the foundation for the robot's understanding of indoor and outdoor scenes, thereby helping the robot to improve its adaptability to the environment and successfully complete the interaction function.
- (4) Other applications. In addition to this, scene point cloud semantic segmentation and 3D object classification also have a wide range of applications in augmented reality, indoor navigation, medical diagnosis, and many other fields and will play an increasingly important role in human future life. It can be seen that point cloud semantic segmentation and 3D object classification technology have very important research significance and application value.

3.2.2 Indoor Scene Understanding Based on Discovering Object Associations via Scene Graphs

Scene understanding is essentially important to computer vision. It essentially simulates the recognition process of the human visual system and understands the content perceived in the surrounding visual scene by mining various association clues hidden in the complex visual world. This scene understanding process can be effectively integrated and assisted with the semantic scene graph (scene graph, SG), which can depict the objects in the scene and their internal relationships (scene layout) as the nodes and edges of the graph, respectively. Different from pure scene knowledge understanding, the task of generating scene graphs requires organizing unstructured scene knowledge into structured, inferable graph structures.

The scene graph generation task, compared to object detection and classification tasks, explicitly models the visual context at the object level for understanding, and it is a more “advanced” visual task. This object association information, that is, the visual context, has a good effect on solving complex semantic understanding tasks, such as image captioning and visual question answering. In fact, visual context is a powerful inductive bias that connects objects in a specific layout for high-level reasoning. For example, the spatial layout of a “person” on a “horse” helps to determine the relationship of “riding,” and if you want to answer “who is riding a horse?,” this in turn helps to locate the “person.” However, in the task of generating scene graphs, it is hoped to organize the scene context knowledge into a graph, and the process of organizing itself is a very important scene understanding process.

Despite the consensus on the value of visual context, in the exploration process of unstructured scene information, context models are diversified into various implicit methods. Implicit models directly encode surrounding pixels into multi-scale feature maps, such as the effective method of increasing the receptive field proposed by dilated convolution [9], which is suitable for various dense prediction tasks [10, 11]; feature pyramid structure [12], which combines low-resolution context features with high-resolution detailed features to promote object detection with rich semantics. In the exploration process of structured scene information, explicit context models are often used to determine the context clues of instances by connecting objects.

3.2.3 Summary

In summary, whether it is based on the object hierarchy or the scene graph hierarchy, the process of understanding the indoor scene is essentially to use the context information of the environment to extract association clues and enhance the robot’s understanding of unfamiliar environments. The task of environmental understanding plays a decisive role in subsequent high-level tasks such as indoor navigation, task planning, and autonomous driving. The transfer and inferability of scene

knowledge can reduce the possibility of planning failure or the need for secondary exploration in high-level tasks.

From the above understanding of the scene understanding methods at two different levels of understanding, it can be seen that each has its own strengths and faces challenges from multiple aspects. Both methods have received attention from the academic and industrial communities and focus on understanding tasks of different real-time degrees. In the following sections of this chapter, based on specific practices, we will further guide readers to understand the challenges and solutions that scene-understanding algorithms and their practical applications will face.

3.3 Technology Practice

3.3.1 *Case Study: Point Cloud Semantic Segmentation and Classification Based on Convolutional Neural Networks*

3.3.1.1 Task Description

The goal of this practice is to implement a local coordinate system plane convolution that can be executed point by point on the point cloud. This convolution method can cleverly apply mature image semantic segmentation technology to 3D scene point clouds, thereby solving the problem of point cloud semantic segmentation. At the same time, this convolution method can also be used in 3D object classification. The entire process can be summarized as using a self-encoding network model similar to the U-Net structure in the field of image semantic segmentation as the overall architecture of the point cloud semantic segmentation algorithm. In this architecture, the expressive power of the model is enhanced by merging the features of the encoder and decoder. At the same time, in the local coordinate system plane convolution module, the computational load is greatly reduced when extracting features from the point cloud by using depth separable convolution. This practice is to solve the problem that convolutional neural networks cannot be successfully applied to 3D point clouds due to their irregular and unordered characteristics. In response to these problems, this chapter has designed a local coordinate system plane convolution that can be executed point by point on the point cloud. Through the use of this convolution, mature image semantic segmentation technology can be cleverly applied to 3D scene point clouds, thereby solving the problem of point cloud semantic segmentation. At the same time, the local coordinate system convolution is used to extract local features of 3D objects, thereby obtaining more detailed information about 3D point cloud objects, and making certain improvements to the PointNet network structure, thereby constructing a more robust 3D object classification model.



Chapter Knowledge Points

Conditional random field (CRF): A conditional random field is a conditional probability distribution model $P(Y|X)$, which represents a Markov random field of another set of output random variables Y under the condition of a given set of input random variables X .

Multilayer perceptron: Multilayer Perceptron (MLP) is the basic algorithm of Deep Neural Network (DNN). The neural network consists of an input layer (first layer), hidden layer (middle layer), and output layer (last layer). In MLP, the layers are fully connected, the forward propagation algorithm is used to solve the relationship between the outputs of two adjacent layers, the input of a layer is the output of the previous layer, and the back propagation algorithm (BP) is used to solve the relationship matrix W and bias vector b of each layer.

Point cloud features: (1) Sparsity. Point cloud data only exists on the surface of objects. (2) Data missing. Some surfaces are not scanned due to occlusion. (3) Data noise. Caused by the precision of the instrument itself or environmental factors. (4) Non-uniformity. Caused by factors such as the instrument's sampling strategy, relative position, scanning range, etc.

3.3.1.2 Related Work

Traditional point cloud semantic segmentation algorithms mainly use artificially constructed features to describe point clouds. The quality of such feature representation usually depends on the experience of experts in the field, making it difficult to extract deep semantic information from point clouds. In recent years, the development of deep learning technology, especially convolutional neural networks, has led to substantial progress in various tasks in the field of computer vision.

Current Research Status of Point Cloud Semantic Segmentation at Home and Abroad

The use of convolutional neural networks to process two-dimensional images dominates the development of modern computer vision. The key to its success is the effective processing of images based on convolution operations. 2D convolution is defined on the regular grid of the image, which supports efficient convolution operations and ensures the processing effect of deep networks on large datasets with high resolution.

When analyzing large-scale 3D scenes, a direct extension of the above method is to perform voxel-based convolution on a 3D voxel grid. One way to improve the accuracy of voxel grid convolution technology is to add differentiable post-processing, such as conditional random fields (CRF) [13]. However, this

voxel-based method still has significant limitations, including cubic growth in memory consumption and computational efficiency issues caused by the sparse attributes of point clouds. For this reason, voxel-based convolutional neural networks often operate on low-resolution voxel grids, which limits their prediction accuracy. In addition, voxel-based convolution can be combined with Octree to solve these problems. This technology defines convolution on the Octree and can handle slightly higher resolution volumes. However, these are still not enough to efficiently analyze large-scale 3D scenes.

Similarly, some novel deep learning methods for analyzing 3D data have emerged. Although they do not involve large-scale semantic segmentation of the entire scene, they provide interesting ideas.

Yi et al. [14] consider shape segmentation in the spectral domain through cross-model synchronous feature vectors. Masci and Boscaini et al. [15–17] designed convolutional neural networks of Riemann manifolds and used them to learn correspondences between shapes. Sinha et al. [18] analyze the shape of geometric images. Simonovsky et al. [19] extend convolution operators from regular grids to arbitrary graphics and use them to design networks for shape classification. Li et al. [20] introduced a field theory-based detection neural network that respects the potential sparsity of 3D data and uses it for efficient feature extraction. Tulsiani et al. [21] use volumetric primitives to approximately represent 3D models in an end-to-end differentiable framework and use this framework to solve multiple tasks. Maron et al. [22] designed a convolutional neural network that operates on spherical surfaces. Overall, most existing 3D deep learning systems either rely on representations that do not support general scene analysis or have poor scalability.

The drawbacks of voxel-based 3D data analysis methods are obvious. Recent studies suggest that voxel-based 3D data structures are not the most natural form of 3D convolution and propose alternatives such as those based on unordered point sets [23], graph structures [19], and spherical surface structures [22]. Unfortunately, these methods all have their own shortcomings, such as limited sensitivity to local structures or reliance on restrictive topological assumptions.

So far, not much work has applied deep learning methods to unordered point sets. PointNet [23] is a pioneering work that applies deep learning methods to unordered point sets and performs point cloud semantic segmentation tasks.

The basic idea of PointNet is to use a shared multilayer perceptron to learn the spatial encoding features of each point, and then aggregate the features of individual points into global features. Based on such a design, PointNet cannot extract the local structure features of the point cloud. However, the features of the local structure have been proven to be crucial to the success of the convolutional architecture. Convolution takes data defined on a regular grid as input and it is capable of gradually capturing local structural features at increasingly larger scales along the multi-resolution hierarchy. Lower-level neurons have smaller receptive fields, while higher-level neurons have larger receptive fields. In addition, Landrieu et al. [24] introduced the super point graph (SPG) to segment large-scale point clouds.

Current Research Status of 3D Object Classification at Home and Abroad

For the research of 3D object shape classification tasks, some deep learning methods based on convolutional neural networks have emerged. According to the representation of 3D objects, these methods can be divided into the following categories:

- (1) **Methods based on voxel grids.** This method models a 3D object as a function sampled on a voxel grid and defines a 3D CNN on the voxel grid for the analysis of 3D object shapes. However, due to the high memory and computational costs of voxel grids, these voxel-based methods are usually limited to low resolutions like 30^3 .
To reduce the computational cost of voxel-based 3D object classification methods, Graham [25] proposed a sparse 3D CNN, which applies convolution operations to activated voxel grids and only activates neighboring voxels within the convolution kernel. However, as the number of convolutions between two pooling operations increases, this method will become less effective. For convolutional neural networks with deeper layers and more convolution kernel parameters, the computational and storage costs of this method are still high.
- (2) **Methods based on manifolds.** Perform CNN operations on geometric features defined on 3D mesh manifolds. Some methods parameterize the surface of 3D objects into patches or geometric images and sample them regularly, then send the sampled feature images to a two-dimensional convolutional neural network for shape analysis. Other methods extend the convolutional neural network to structures defined by irregular triangular meshes. Although these methods are robust to isometric transformations of 3D shapes, they are all constrained to smooth manifold meshes. And the local features of the 3D objects used in these methods are computationally expensive. A good summary of these techniques can be found in the articles by Bronstein et al [26, 27]
- (3) **Methods based on multiple views.** This method renders 3D objects from different perspectives with a virtual camera, and uses the set of images obtained to represent the 3D object, then uses this set of images as the input to the two-dimensional convolutional neural network to analyze the shape of the 3D object. Although these methods can directly use image-based convolutional neural networks for 3D shape analysis and handle high-resolution input data, it is currently unclear how to determine the number of views, whether the allocated views can cover the 3D object, and how to avoid self-occlusion are also problems encountered by this method.
- (4) **Methods based on point clouds.** PointNet [23] is one of the earliest network architectures that can handle point clouds and can be used for 3D object classification. When using it, only minor modifications to its semantic segmentation network are needed. PointNet is very powerful because it can use order-invariant functions to normalize the input point cloud, but it does not consider the correlation between points when extracting features from the point cloud. To solve this problem, PointNet++ [1] improved it with the farthest point sampling algorithm and neighborhood query algorithm.
- (5) **Graph-based methods.** Recently, Qi et al. [28] proposed a graph neural network for 3D object classification on point clouds, where each graph node is a set of

points, and the edges of the graph are constructed by nearest neighbor search on the point cloud. Their results are displayed with RGB-D images [29]. The parameter weights of the VGG-16 network used for image classification are used to initialize the prediction.

3.3.1.3 Technical Methods

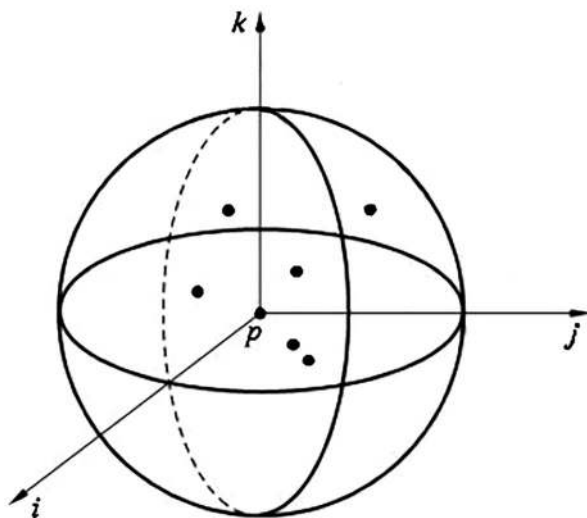
In order to perform convolution on point clouds in a point-by-point manner, it is necessary to use the principal component analysis (PCA) to estimate the local coordinate system of the point cloud, obtain the parameters of the local coordinate system plane, then project the neighboring points of the point cloud, calculate the projection position, and finally calculate the signal quantity at the corresponding position of the convolution kernel in the local coordinate system plane through the nearest neighbor interpolation algorithm.

Local Coordinate System Plane Convolution

Assume P represents the point cloud, representing some kind of signal quantity about P , such as color, normal vector, coordinate information, or abstract features from the middle layer of the network. In order to extract features on the point cloud P using convolution, the point cloud needs to be transformed in a certain way. Here, the point cloud is projected onto three two-dimensional planes, which refer to the three planes of the local coordinate system of the point cloud. These three planes are represented as virtual images with continuous signals, as shown in Fig. 3.1.

As shown in Fig. 3.2, for any point p (red) in the point cloud, its local coordinate system plane parameters can be estimated through local covariance analysis.

Fig. 3.1 Local coordinate system



Specifically, for a set of points q (blue) in a spherical neighborhood satisfying $|p-q| < R$, the direction of the local coordinate system plane of point p is determined by the eigenvectors of the covariance matrix $\sum_q rr^T$, where $r = q-p$. First, all r^i are decentered, i.e.,

$$r^i = r^i - \sum_{j \in q} r^j \quad (3.1)$$

Then calculate its covariance matrix XX^T , perform eigenvalue decomposition on the covariance matrix XX^T , calculate the three eigenvectors i, j, k corresponding to different eigenvalues, and normalize them. Taking the tangent plane as an example, the direction of the tangent plane is the eigenvector k corresponding to the smallest eigenvalue, and the other two eigenvectors i, j here serve as the directions of the two coordinate axes of the tangent plane, and also the normal directions of the other two local coordinate system planes. Note that since the eigenvalues corresponding to the eigenvectors i, j, k of the three local coordinate system planes are arranged in descending order, the structure information of the point cloud preserved in the tangent plane formed by the two directions i, j is the richest, and the information preserved by the other two local coordinate system planes decreases in turn.

Suppose the resolution of the local coordinate system plane is set to 3×3 , then there are 9-pixel points and their features on each local coordinate system plane that will participate in the convolution calculation. The goal of signal interpolation here is to use the set q and its signal quantity $F(q)$ to estimate the signal quantity $T(u)$ at the 9 positions participating in the convolution operation in the local coordinate system plane, because q may be projected to any position in the local coordinate system plane of p , and the 9 positions participating in the convolution operation in the local coordinate system plane are fixed. Still taking the tangent plane as an example, here first map q to the tangent plane of p , which generates a set of projection points $h = (r^Ti, r^Tj)$, as shown in Fig. 3.2. For the signal quantity at h , it is defined here:

$$T(h) = F(q) \quad (3.2)$$

At this time, h is scattered at any position in the image plane. Therefore, these signals need to be interpolated in a certain way to estimate the signals at the 9

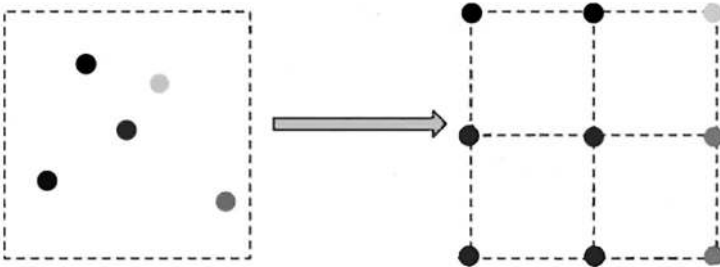


Fig. 3.2 Nearest Neighbor Interpolation (see color picture before the text)

positions involved in the convolution operation on the local coordinate system plane. The mathematical expression is as follows:

$$T(u) = \sum_h w(u, h) \bullet S(h) \quad (3.3)$$

where $w(u, h)$ is the weight of interpolating the signal at u on the plane using the signal at h , and satisfies $\sum_h w = 1$. Here, a relatively simple interpolation method is used—nearest neighbor interpolation. In this interpolation strategy, if $l \in h$ is the nearest neighbor of u , then $w(u, h) = 1$, otherwise $w(u, h) = 0$.

Finally, the formula for convolution on the tangent plane at point p is given:

$$G(p) = \int_{\pi_p} W(u) \cdot \sum_h (w(u, h) \bullet S(h)) du \quad (3.4)$$

where $W(u)$ represents the weight of the convolution kernel at coordinate u on plane π_p . Note that when performing convolution operations on the two local coordinate system planes outside the tangent plane, depending on the projection position, only the $w(u, h)$ in the above formula, i.e., the weight of plane interpolation, needs to be modified. This allows this method to support the construction of deep networks on point clouds with millions of points.

Point Cloud Semantic Segmentation Network

Since the task of point cloud semantic segmentation is to predict the semantic category of each point, the main challenge of this task is how to extract sufficiently expressive features for each point, so that the model can achieve high accuracy while making the boundaries of the predicted results clearer.

As shown in Fig. 3.3, where the yellow part represents the input, i.e., the scene point cloud, the blue part represents the convolution block, the orange part represents the output, and the gray part represents the copied features obtained through

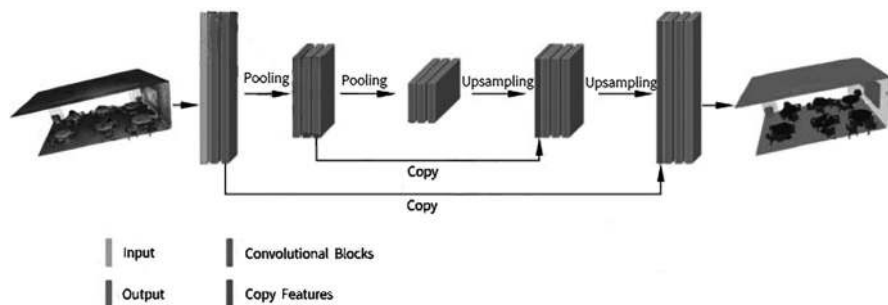


Fig. 3.3 Point cloud semantic segmentation network structure (see color picture before the text)

skip connections. This is a self-encoding network structure similar to U-Net. The encoder is used to extract features from the input point cloud, i.e., to abstractly represent the input features of the point cloud. The encoder has two pooling layers, so during the encoding phase, the receptive field of the point cloud features gradually increases. The decoder infers the semantics of the point cloud using the features of the encoder, and correspondingly, there are two upsampling layers in the decoder to gradually restore the number of point clouds. It can be seen that there are two skip connections between the encoder network and the decoder network. This design allows the decoder to use not only the deep semantic features of the network when predicting the semantics of the point cloud, but also the texture features of the shallow point cloud of the encoder, which is very helpful for predicting the contours of objects. The input and output sizes of the network are the same, and the entire network is also an end-to-end model.

The design of the convolution block is shown in Fig. 3.4, where (1) is to convert the feature vector into the form of local coordinate system plane convolution; (2) is to concatenate the $n \times 3 \times 3 \times d$ tensor on each face into a $n \times 3 \times 3 \times 3d$ tensor; (3) is depth separable convolution; (4) is 1×1 convolution, also known as point-wise convolution here; (5) is to compress the dimensions of the tensor.

The design of such convolution blocks not only allows for feature extraction but also solves the problem of redundant features. When using local coordinate system plane convolution, the point cloud is projected onto the three planes of the local coordinate system, causing the point cloud information in the three directions of the local coordinate system to appear twice, meaning the features obtained at this time are redundant. Depthwise separable convolution, in addition to feature extraction, can also be used for information decoupling. This is because when using depthwise separable convolution, there is no connection between the information of any two channels in the feature map. Therefore, this convolution method will be used in each convolution block in this paper. Depthwise separable convolution actually uses n convolution kernels to process the n channels of the feature map separately, so another advantage of using depthwise separable convolution is that it can significantly reduce network parameters.

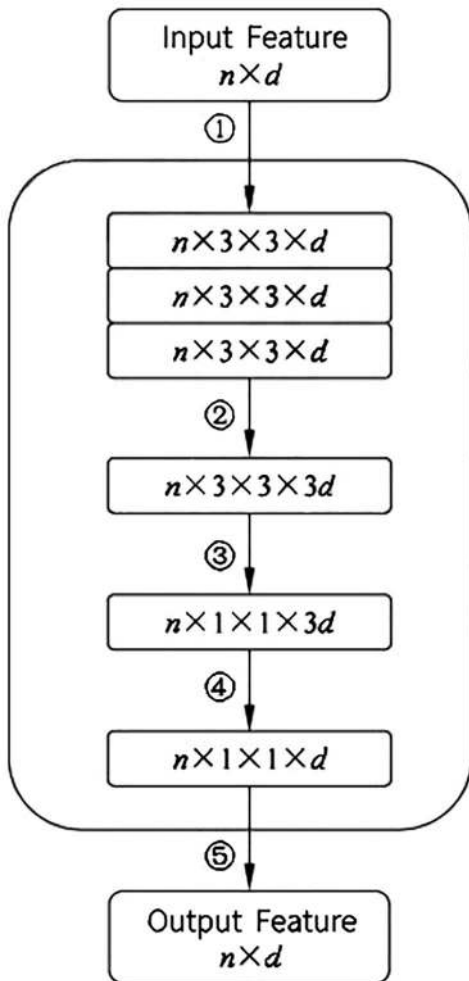
Specifically, for standard convolution, assuming the size of the input feature F is (D_F, D_F, M) , the convolution kernel K is (D_K, D_K, M, N) , and the size of the output feature G is (D_G, D_G, N) , its calculation formula is

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \quad (3.5)$$

where the number of input channels is M , and the number of output channels is N . The total amount of computation is: $D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$.

When using depthwise separable convolution, the standard convolution kernel of shape (D_K, D_K, M, N) can be decomposed into a depth convolution kernel of shape $(D_K, D_K, 1, M)$ and a pointwise convolution kernel of shape $(1, 1, M, N)$. The role of the depth convolution kernel is to filter the input feature F , generate the output feature $\hat{G}$, size (D_G, D_G, M) , its formula is

Fig. 3.4 Convolution block structure



$$\hat{G}_{k,l,n} = \sum_{i,j} \hat{K}_{i,j,m} F_{k+i-1,l+j-1,m} \quad (3.6)$$

where $\hat{K}$ is the convolution kernel of depth convolution, the m th channel of the convolution kernel only processes the m th channel of F , and produces the m th channel feature of $\hat{G}$. The role of the pointwise convolution kernel is to convert the number of channels of features, process $\hat{G}$, and obtain the final output of shape (D_G, D_G, N) . Therefore, the total amount of computation of depthwise separable convolution is $D_K \cdot D_K \cdot M \cdot D + D_F \cdot D_F \cdot M \cdot N$.

Comparatively, the ratio of the amount of computation of depthwise separable convolution to the amount of computation of standard convolution is

$$\frac{D_F \cdot D_F \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \quad (3.7)$$

Here, $D_K = 3$, N is 32, 64, 128, etc., so the value of Eq. (3.7) is about $\frac{1}{9}$. Therefore, using depthwise separable convolution here can significantly reduce the amount of computation when using local coordinate system plane convolution.

Convolutional neural networks often use pooling layers to aggregate signal quantities over larger spatial areas. Scatter the point cloud onto a 3D voxel grid, and achieve the pooling layer by continuously reducing the resolution of the voxel grid. In the process of voxel grid resolution change, every eight smaller voxel grids will merge into a larger voxel grid, and at the same time, pooling operation is performed on the points and their features in the eight smaller voxel grids to generate a new point and its new feature. This process is shown in Fig. 3.5.

In Fig. 3.5, the resolution of the grid changes from $2 \times 2 \times 2$ to $1 \times 1 \times 1$, and the 7 points before pooling become 1 new point after pooling. Let $P = \{p\}$ represent the set of points before pooling, $F\{p\}$ represent the signal quantity of each point, p' represent the new point generated after pooling, then the coordinates of this point are

$$p' = \frac{1}{|P|} \sum_{p \in P} p \quad (3.8)$$

By using average pooling, the signal quantity of this point becomes

$$F(p') = \frac{1}{|P|} \sum_{p \in P} F(p) \quad (3.9)$$

In this way, the signal quantities of multiple points in the point cloud will be gathered to a new point, which can enhance the ability of network abstraction expression and at the same time reduce the amount of computation. Note that when using the pooling layer for downsampling, the process is independent of the signal quantity of the point cloud, but only related to the spatial distribution of the point cloud. Therefore, the three pooling layers in the network model can be calculated in preprocessing, obtaining the

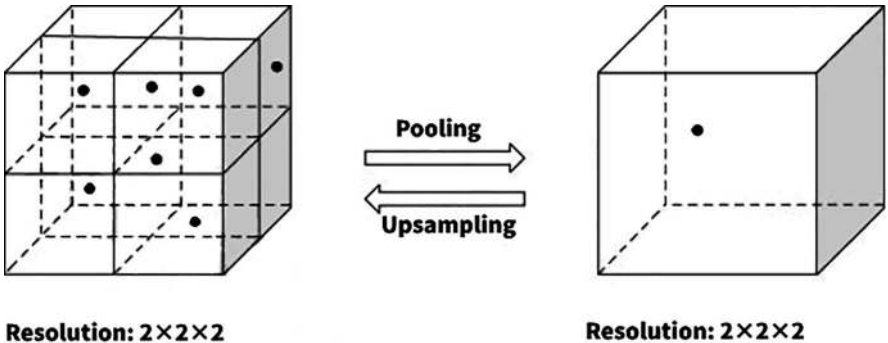


Fig. 3.5 Pooling and upsampling

point cloud after each pooling layer, the plane parameters of the local coordinate system of the point cloud after pooling, and the indices required for pooling.

The specific implementation of the pooling layer is similar to the implementation of the local coordinate system plane convolution. Suppose the input of a pooling layer is F_{in} , with a size of $N_{in} \times C$, where N_{in} is the number of points in the point cloud before pooling, and C is the number of channels of the point cloud features. The purpose of the pooling layer is to downsample the current point cloud and get the output F_{out} of $N_{out} \times C$, where N_{out} is the number of points in the point cloud after pooling, and it satisfies $N_{in} > N_{out}$.

Specifically, in the preprocessing stage, the point cloud will be first downsampled to a voxel grid of size 5^3 cm, so at this time, there will be at most one point in each voxel grid. Then for the input point cloud $P = \{p\}$, a new point cloud $P' = \{p'\}$ is generated by using voxel grid sampling, and a pooling index matrix M of size $N_{out} \times 8$ is obtained, where $N_{in} = |P|$, $N_{out} = |P'|$. Note that because 8 small voxel grids correspond to 1 large voxel grid during pooling, and there is at most one point in each small voxel grid, the new point generated after pooling corresponds to at most 8 original points. In the operation phase of the network, for the current feature F_{in} , an intermediate variable of size $N_{out} \times 8 \times C$ can be generated by using the pooling index matrix M calculated in the preprocessing stage, and then the feature map F_{out} of $N_{out} \times C$ is obtained by calculating formula (3.9).

The upsampling layer has the opposite effect to the pooling layer. Here, the upsampling layer restores the number of points in the point cloud by increasing the resolution of the voxel grid. In the process of voxel grid resolution change, a larger voxel grid will be decomposed into eight smaller voxel grids, and at the same time, the features of the points in the low-resolution voxel grid will be copied to each point in the high-resolution voxel grid. There are two upsampling layers in the network model. In this process, the number of points in the point cloud is restored from 50,000 to 200,000. The pooling method is shown in Fig. 3.5.

Specifically, suppose the input of an upsampling layer is F_{in} , with a size of $N_{in} \times C$, where N_{in} is the number of points in the point cloud before upsampling, and C is the number of channels of the point cloud features. The upsampling layer maps the points in the low-resolution voxel grid by using the index matrix M calculated in the pooling layer, copies its features to the points in the higher-resolution voxel grid, and thus obtains the output F_{out} of $N_{out} \times C$, where N_{out} is the number of points in the point cloud after upsampling, and it satisfies $N_{in} < N_{out}$.

Note that in addition to the features copied from the low-resolution point cloud in the upsampling layer, some texture features of the lower layer of the network will also be obtained through the skip connections in Fig. 3.3. Therefore, the upsampling layer needs to concatenate these two parts of features to get the real output of this layer.

3D Object Classification Network

This section designs the network architecture shown in Fig. 3.6 for classifying 3D models, based on the characteristics of 3D objects and inspired by the image classification network VGG-Net and the first model PointNet that directly processes point clouds.

As shown in Fig. 3.6, the entire network structure consists of four modules, including local coordinate system plane convolution, spatial transformer network module, max pooling layer, and fully connected layer. The input of the network is a 3D object represented by a point cloud. The size of the input is adjusted to $1024 \times m$ by randomly sampling the surface of the 3D object in the sampling layer, where m represents the channel number of the input point cloud features. These features usually include color information, normal vector information, and coordinate information. Here, only the x, y, z coordinate information of the point cloud is used, so $m = 3$ in this case. The network has two spatial transformer networks (STN) modules and five convolution layers. The two STN modules output an $m \times m$ and a 64×64 size matrix, respectively, which are used to spatially transform the input features and intermediate layer features of the network, thereby increasing the rigidity invariance of the network. The usage is as follows:

$$F_{\text{out}} = F_{\text{in}} \bullet M \quad (3.10)$$

where F_{in} is the input feature of the STN module, M is the transformation matrix calculated by the STN module, and F_{out} is the output result of the spatial transformation of F_{in} . Each convolutional layer is a localized coordinate system planar convolution that can be performed on a point cloud, as proposed earlier in this chapter. The convolution kernel size of each convolution layer is 3×3 . The number of convolution kernels contained in each convolution layer are 64, 64, 64, 128, and 1024, respectively. A batch normalization layer is added after each convolution layer to improve the model effect. Its mathematical form is as follows:

$$y = \frac{\gamma}{\sqrt{\text{Var}[x] + \varepsilon}} \cdot x + \left(\beta - \frac{\gamma E[x]}{\sqrt{\text{Var}[x] + \varepsilon}} \right) \quad (3.11)$$

where x is the feature value of a certain channel of the convolution layer, γ and β are the scale and shift parameters obtained from training, $E[x]$ is the global expected value of x , and $\text{Var}[x]$ is the global variance value of x .

After the convolution layer, a max pooling layer is used to provide a summary representation of the features of the last convolution layer. Note that to solve the

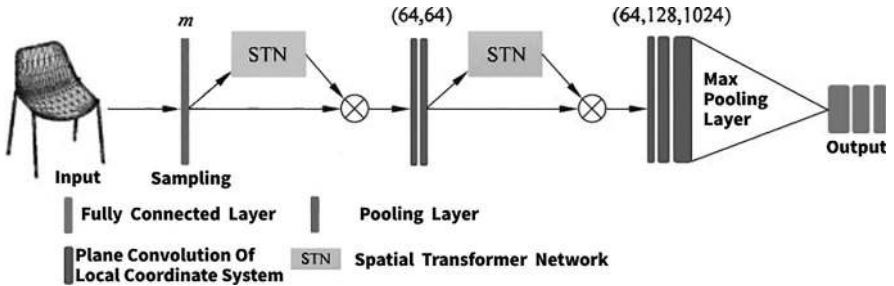


Fig. 3.6 3D object classification network structure (see color figure before the text)

problem of point disorder, this max pooling layer is executed simultaneously for 1024 points, so the input and output of this max pooling layer are 1024×1024 and 1×1024 , respectively. The tail of the network has two fully connected layers, containing 512 and 256 neuron nodes, respectively. The last layer of the network is a normalized exponential function layer, which maps the features of the last connection layer to obtain a score vector. In the experiment, the number of score vectors is set to the number of object categories in the dataset.

During the training phase of the network, it is first necessary to initialize the parameters of various parts of the network, then input the 3D point cloud and its true category at the input end, and then the network forward propagation obtains a 256-dimensional global feature of the 3D object. The cross-entropy is used as the loss function to obtain the training error and then implement backpropagation to optimize and update the parameter values of various parts of the network, so that the accuracy of the network's judgment can be continuously improved. The formula for cross-entropy loss is

$$L = -\frac{1}{n} \sum_i^n \sum_j^k y_{ij} \log(p_{ij}) y \quad (3.12)$$

where $\log$ is based on e , for sample x_i , y_i is its true category, $y_{ij} = 1$ if and only if $j = y_i$, otherwise $y_{ij} = 0$. p_{ij} is the probability of predicting sample x_i as the j th label, and n is the total number of samples. In the testing phase, for the input 3D model, only the forward propagation of the network is performed to obtain the predicted category of the model.

Since the input of the network is a 3D point cloud, the traditional 2D convolution in the image cannot be used directly. Therefore, the local coordinate system plane convolution proposed earlier that can be used on the point cloud is used here to extract features from the point cloud. This method can map the local structural information of the point cloud to a two-dimensional plane, so combined with coordinate interpolation and other methods, the point cloud can be transformed into a two-dimensional convolution form.

Here, let's discuss the advantages of using local coordinate system plane convolution for feature extraction compared to the use of multilayer perceptrons in PointNet. It should be noted that the task of semantic segmentation of point clouds requires predicting the semantic category of each point, thus it requires more emphasis on the expression of local features of point clouds and sufficient feature extraction of the structural information of each point. The classification task of 3D objects only requires the network model to make a judgment on the category of the 3D object, thus it focuses more on the global features of the 3D object. Therefore, when using local coordinate system plane convolution in the classification task of 3D objects, only the neighborhood information of each point is projected onto the tangent plane. Although this will lose some information of the point cloud, it actually compresses the network model to a certain extent and does not affect the performance of the network.

In PointNet, for the input model point cloud, each point is processed individually using a multilayer perceptron. Obviously, using this method to extract features only

treats the point cloud as a collection of isolated points, but does not consider the connection between points. Therefore, compared to multilayer perceptrons, local coordinate system plane convolution has the following advantages:

- (1) As is well known, in the two-dimensional convolution in images, when extracting features for each pixel, the information of the neighboring pixels covered by the convolution kernel is combined, and then through the training of the convolution kernel weights, the deep learning algorithm can learn some kind of connection between pixels. The features extracted in this way can fully utilize the spatial relationship and semantic association between pixels. When using local coordinate system plane convolution in point clouds, it is not processing each point individually, but processing the points in each sphere centered on each point. In this way, the spatial relationship between points in three dimensions is explicitly expressed.
- (2) Using local coordinate system plane convolution can increase the receptive field of features. The receptive field is an important concept in convolutional neural networks. It refers to the size of the input layer element area related to the output result of a layer in the convolutional neural network. The calculation formula for the receptive field is

$$RF_n = RF_{n-1} + (k_n - 1) \times \text{stride}_n \quad (3.13)$$

where RF_n is the receptive field of the features of the n th layer of the network, k_n is the size of the convolution kernel of the n th layer of the network, and stride_n is the stride of the convolution layer of the n th layer of the network. It can be seen that when k_n is greater than 2 and stride_n is equal to 1, the receptive field of the n th layer of the network will still increase. Therefore, in the work of this chapter, although the receptive field of the first layer of the network is the size of the sphere in the initialization, because the size of the convolution kernel used in the local coordinate system plane is 3, the receptive field of the network will still increase with the increase of the number of network layers.

As mentioned earlier, because point clouds are a form of unordered 3D data structure, when designing the network, the network needs to be able to adapt to the unordered input of point clouds, that is, the order of point cloud data input should not affect the prediction results of the network. Here, the method of maximum pooling is used to map the local features of the point cloud to global features, thereby solving the problem of point cloud disorder. That is, for the features of the point cloud obtained from the last convolution layer $\{x_1, x_2, \dots, x_{1024}\}$, where $x_i \in R^{1024}$, its global features can be calculated by Eq. (3.14), that is,

$$y^i = \max(x_1^i, x_2^i, \dots, x_{1024}^i) \quad (3.14)$$

where $y \in R^{1024}$ represents the global features of the sample, that is, the global features of the sample are actually the results of maximum pooling in each channel of the local features of the point cloud. After obtaining this global feature, a classifier for 3D objects can be trained using a multilayer perceptron and a cross-entropy function.

Spatial transformer networks (STN) are first used in image processing, because when convolutional neural networks (CNN) are used for image classification and similar problems, they often need to consider the scale invariance, rotation invariance, and translation invariance of the image. Overcoming these issues can often improve the accuracy of the algorithm. The essence of these invariances, which involve image rotation, translation, and scaling, is to perform spatial transformations on the image.

The affine formula for the image is as follows:

$$\begin{bmatrix} x_t & y_t \end{bmatrix} = \begin{bmatrix} x_s & y_s & 1 \end{bmatrix} \begin{bmatrix} \theta_{11} & \theta_{21} \\ \theta_{12} & \theta_{22} \\ \theta_{13} & \theta_{23} \end{bmatrix} \quad (3.15)$$

Here, x_s and y_s represent the coordinate values of the pixel points in the original image, x_t and y_t represent the coordinate values of the pixel points after the affine transformation, and the matrix θ in the formula is the affine transformation matrix. This matrix can be used to perform rigid transformations such as scaling, translation, and rotation on the image.

The problem that the spatial transformer network needs to solve is to adaptively obtain a certain affine transformation matrix, and then perform spatial transformation and alignment of the data in the middle layer of the network as required by the task. In this way, even when the input sample data space distribution has a large difference, the model trained still has high accuracy.

In this chapter, the STN module is also added to learn a transformation matrix that can help the network complete the classification task. The network structure diagram is shown in Fig. 3.7:

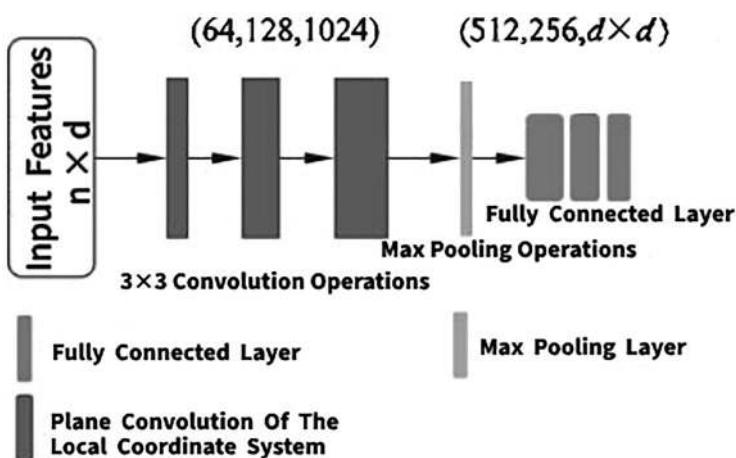


Fig. 3.7 STN network structure (see color picture before the text)

For the input feature map of size $n \times d$, where n is the number of points and d is the current number of feature channels of the point cloud, the role of the STN network here is to solve a spatial transformation matrix $\mathbf{M}$ of size $d \times d$ for this feature map. The STN network used in this chapter contains 3 convolutional layers, 3 fully connected layers, and the 3 convolutional layers contain 64, 128, and 1024 convolution kernels, respectively. A batch normalization (BN) layer is added after each convolutional layer, and after the third convolutional layer, max pooling operation is also required for each channel of the feature to obtain a global 1024-dimensional feature. The tail of the STN network adds 3 fully connected layers, which contain 512, 256, and $d \times d$ neurons, respectively, to infer the parameters of the transformation matrix.

Two STN modules are used here. The first time is to directly adjust the 3D coordinate space of the input point cloud, which can be understood as allowing the network to find an angle that is conducive to model classification. For example, it can be assumed that the model is rotated to the front through the addition of this module. The second time is to adjust the features in the middle layer of the network, which is to transform the point cloud at the feature space level. During training, the coefficients of the two transformation matrices are learned and optimized by controlling the loss function of the network.

3.3.1.4 Results Display

For the point cloud semantic segmentation experiment, the S3DIS dataset, i.e., Stanford 3D Indoor Semantics Dataset, is used. This is a large indoor scene dataset from Stanford University, which includes six large indoor areas of three different buildings, including offices, conference rooms, restaurants, kitchens, bathrooms, warehouses, and other types of scenes, as shown in Fig. 3.8. Each file contains tens of thousands of points, each point's information is composed of coordinates and color, etc. In addition, the data includes 13 object categories. In the experiment, this book uses Area 5 for testing and the rest for training.

For the point cloud object classification experiment, the dataset used is ModelNet40, which contains 12,311 CAD models. As shown in Fig. 3.9, each model varies in size and orientation, and there are 40 semantic labels in total, mainly including common object categories such as tables, chairs, desk lamps, beds, etc. In the experiment, 9843 CAD models were used for network training, and the remaining 2468 models were used for testing.

3.3.1.5 Summary

In this chapter, a local coordinate system plane convolution that can be used on point clouds is proposed. This convolution achieves the conversion of 3D information to two dimensions by projecting the point cloud onto the local coordinate

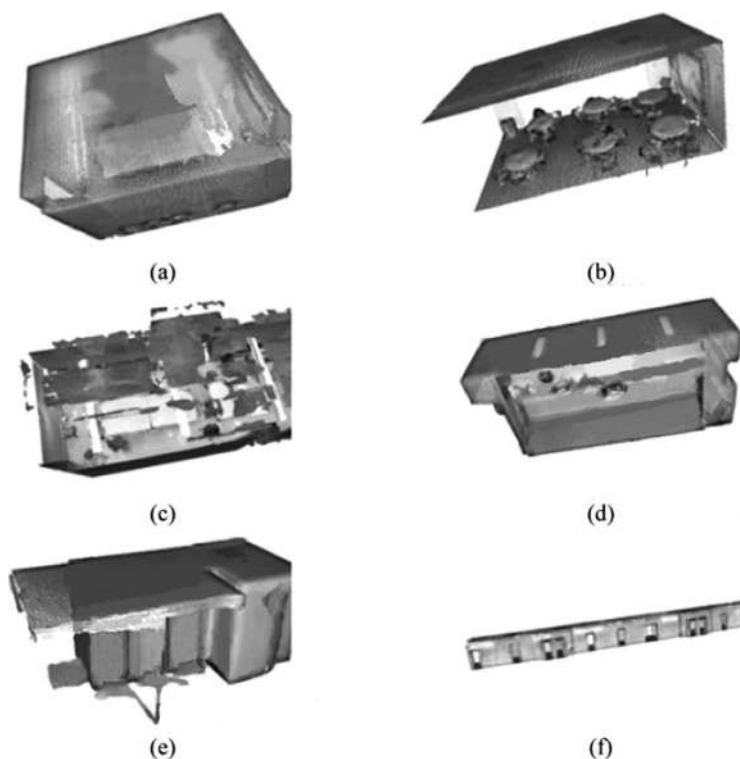


Fig. 3.8 Semantic segmentation results. (a) Office; (b) Restaurant; (c) Conference Room; (d) Kitchen; (e) Bathroom; (f) Warehouse

system plane, thus enabling point-by-point convolution and abstract learning of local features of the point cloud. It predicts the semantic categories of the point cloud end-to-end using a network structure similar to the typical U-Net in the field of image semantic segmentation. In addition, this algorithm designs a neural network architecture for 3D object classification, considering the characteristics of point clouds such as disorder, rigid transformation invariance, and structural association between points. This network architecture uses local coordinate system plane convolution to automatically learn the deep features of point clouds, thereby achieving an abstract representation of 3D objects, and solves the impact of point disorder on the application of deep learning by performing maximum pooling on all points simultaneously. Experimental results show that compared with existing 3D object classification algorithms, this method can predict the categories of 3D objects more efficiently.



Fig. 3.9 Part of the ModelNet40 dataset's 3D objects

3.3.2 Case Study: Scene Graph Generation Based on 3D Scene Point Clouds

3.3.2.1 Task Description

Since the quality and completeness of 3D point cloud data are significantly different from two-dimensional images, there are significant problems with directly migrating two-dimensional scene graph generation methods to three dimensions. Existing 3D scene graph generation methods directly borrow common visual patterns and context information fusion methods from two-dimensional methods, resulting in a lot of noise in the extracted features and gradually losing discriminability in the subsequent context information fusion process. This greatly reduces the robot's understanding ability in 3D scenes and will limit the implementation effect of subsequent advanced tasks, such as robot navigation, task planning, and scene modification tasks.

The goal of this practice is to implement a scene graph generation method for 3D scene point cloud data with instance segmentation labels. In the generation process, the robot needs to learn not only the labels of each instance in the scene but also the relationship labels between instances (including spatial relationships, comparative relationships, and belonging relationships). Using the scene graph skeleton without node labels, object node labels of different granularities, and relationship labels as supervision data to implement the 3D scene graph generation method, realizing the structured scene knowledge extraction process.



Chapter Knowledge Points

Scene understanding: Scene understanding is the process of perceiving, analyzing, and interpreting the observed 3D dynamic scene through sensor networks, which is usually real-time.

Unstructured data: Unstructured data is data with irregular or incomplete structure, no predefined data model, and not easy to represent with a two-dimensional logical table of a database. It includes all formats of office documents, text, images, XML, HTML, various reports, images, and audio/video information, etc.

Structured data: Structured data, also known as quantitative data, is information that can be represented with data or a unified structure, such as numbers, symbols.

Semantic scene graph: By depicting the objects in the scene and their internal relationships (scene layout) as the nodes and edges of the graph. Respectively, a structured description of the content in the scene is achieved.

3.3.2.2 Related Work

Scene graph (SG) was first introduced into computer vision to capture more semantic information of objects and their interrelationships for image retrieval [30]. Subsequently, Xu et al. [31] used gated recurrent units to iteratively propagate messages between the original graph and the dual graph formed by scene graph nodes and edges, while MotifNet [32] generated scene graphs using global context information parsed by bidirectional LSTM. Most methods have solved the object prediction problem in the scene graph within the core framework of the object detector, used for specific feature extraction of nodes and edges, while Graph R-CNN [33] proposed an attention variant of graph convolution and combined it with the object detector to handle the context information between objects and relationships.

In 3D, SG has only recently become more popular with the introduction of the 3DSSG dataset. Due to the data quality and completeness of 3D point clouds, there are usually significant problems with directly migrating scene graph generation methods on 2D images to 3D point clouds, as the noise from the data level is more than that of 2D images. From the implementation perspective, scene graph generation methods on 2D images can be divided into two categories: the process of understanding the environment with prior knowledge, and the process of organizing scene knowledge based on environmental context. At present, several types of prior knowledge have been proposed. The traditional prior knowledge is a message-routing method based on the co-occurrence of objects and relationships, but due to its dependence on the prediction quality of instances, it has fragile fault tolerance for the classification results of objects.

Another direction aims to explore scene knowledge by exploring different message-passing mechanisms. However, in 2D methods, because the data quality is very high, the methods operate on environmental context and noise

indiscriminately, leading to feature confusion. Although existing 3D scene graph generation methods directly input 3D scene point clouds with instance segmentation labels, they still cannot solve the impact of noise on feature recognition. Therefore, how to construct discriminative features based on feature extraction methods and environmental context fusion is an important research content.

3.3.2.3 Technical Methods

SGG methods first need to define the visual patterns of objects and relationships in the implementation process. Traditional methods will use the pixels or point cloud data within the object candidate box as the visual pattern of the object, while the visual pattern of the relationship has a variety of spatial definitions, such as the union box (the union of two object boxes), intersection box, and vector space (the directed vector space formed by the center points of two object boxes). However, none of these three are suitable for the data form of 3D scene point clouds. The union box, due to repeatedly modeling the object space, constitutes a large amount of information redundancy, while the intersection box and vector space, due to covering too few data points, result in insufficient environmental information. Moreover, because 3D data has a lot of noise, therefore, the use of visual patterns defined by two-dimensional SGG methods directly in 3D SGG methods will cause excessive information redundancy, making it impossible to extract effective information from the environment sufficiently.

Using the defined visual patterns to extract initial features, the method of environmental context fusion is used to increase the discriminability of features. The relatively similar initial features, by integrating environmental context information from different regions, become distinguishable, achieving precise prediction for objects and relationships. Traditional SGG methods mostly use the message-passing (MP) mechanism to achieve this process. In traditional methods, the MP mechanism is often built on a fully connected graph, indiscriminately passing all initial features to each other. This way of passing causes the noise information contained in the features to be repeatedly superimposed, causing feature confusion and reducing the accuracy of entity recognition.

Therefore, based on the investigation and analysis of existing scene graph generation methods, a new visual model suitable for 3D point cloud data, namely, the interaction space, is designed to extract initial features with moderate redundancy. This reduces the noise accumulation problem caused by repeated modeling while ensuring sufficient environmental information. At the same time, the process of understanding mimics the human process of environmental understanding, designing an environmental context fusion process with structured organization and hierarchical processing. By transforming the indiscriminate information transmission process into a structured information organization process, the construction of features with higher discriminability is achieved, realizing a more accurate 3D scene graph generation task.

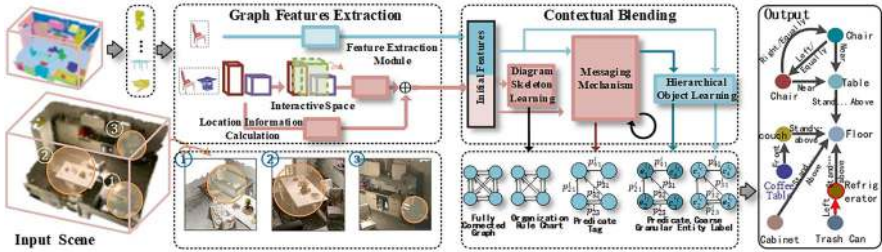


Fig. 3.10 Scene graph generation process based on scene point cloud (see color figure in front of the text)

The overall process takes a scene point cloud with instance segmentation point cloud labels as input, and extracts initial features with moderate redundancy through a new visual model suitable for 3D point cloud data. In the environmental context fusion process with structured organization and hierarchical processing, the construction of features with higher discriminability is achieved, realizing a more accurate 3D scene graph generation task. The specific module design is shown in Fig. 3.10.

By dividing the scene into several small scenes, each small scene contains 4–9 entities, and then the scene graph of the whole scene is divided into small graphs; the point cloud contained in each object in the scene is divided according to the instance segmentation label, and the object box of the object is obtained by calculating the minimum and maximum values of the point cloud.

First, it is necessary to judge whether the two objects are intersecting or separated based on the object boxes of the two objects. If the two objects intersect, the intersection area is the interaction space; if the two objects are separated, the center points of the two object boxes are selected as the lower left corner and upper right corner of the interaction space, respectively, and the length, width, and height of the interaction space are the differences between the two point coordinates, as shown in Fig. 3.11.

The scene point cloud within the object box is defined as the visual pattern of the object, and the scene point cloud within the interaction space is the visual pattern of the relationship. The scene point clouds of the object and the relationship are sequentially input into the PointNet feature extraction model to obtain the initial features of the object and the relationship. The positioning feature of the object is calculated, the union space is calculated according to the object boxes of any two objects, and the position feature of the object is obtained by subtracting the minimum and maximum points of the subject-object box and the object box from the minimum and maximum values of the union space, respectively. The fusion of position features and relationship initial features can effectively supplement the position information of the relationship initial features.

The graph skeleton learning module consists of three fully connected layers, with the obtained relationship initial features as input, and the skeleton without

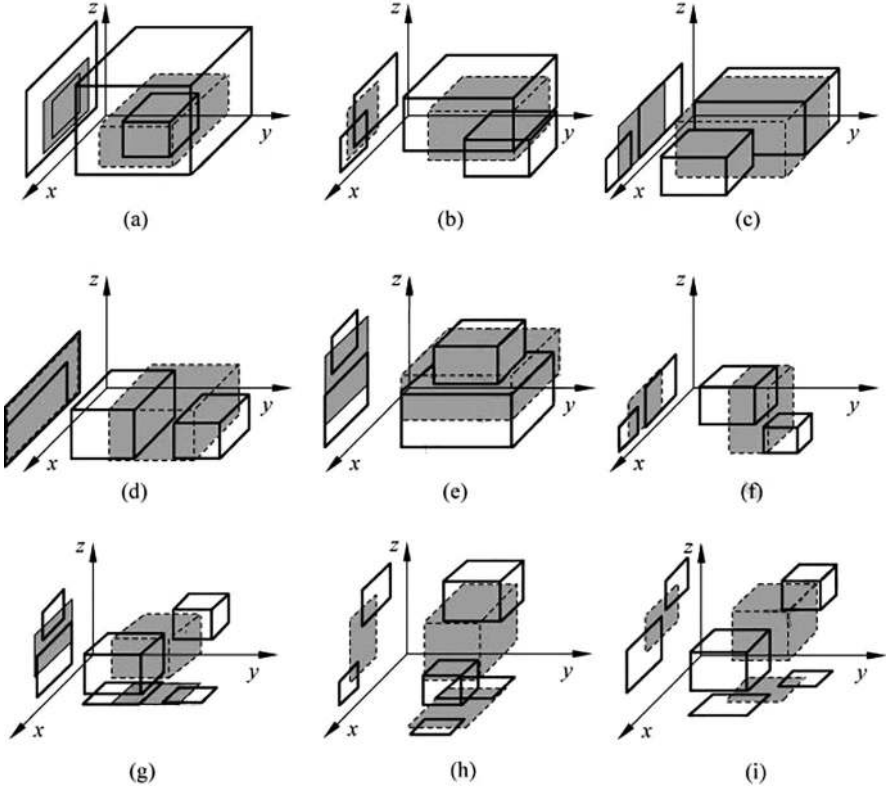


Fig. 3.11 Interaction space definition types

node and relationship labels as supervision data, to predict the association degree between any two nodes in a binary classification. The graph skeleton learning module predicts a probability value for the connection status of every two nodes, normalizes it through Softmax, and generates the organization weight for the subsequent message-passing mechanism.

The environmental context fusion module mainly uses the message-passing mechanism MP to achieve, and the input of MP includes the organization weight calculated by the graph skeleton learning module, and the initial features of the object and the relationship. MP mainly uses the bipartite graph neural network (BGNN) to achieve, BGNN includes two parts, the message passing from entity to relationship and the message passing from relationship to entity, by multiplying the organization weight learned by the graph skeleton learning module with the two parts of the message, the combination of the message passing process and the structural information is achieved, and then the differential information transmission is realized by using the weight. Through multiple cycles, the process of integrating environmental context information in the initial features of objects and relationships is realized, and the fine features of objects and relationships are output.

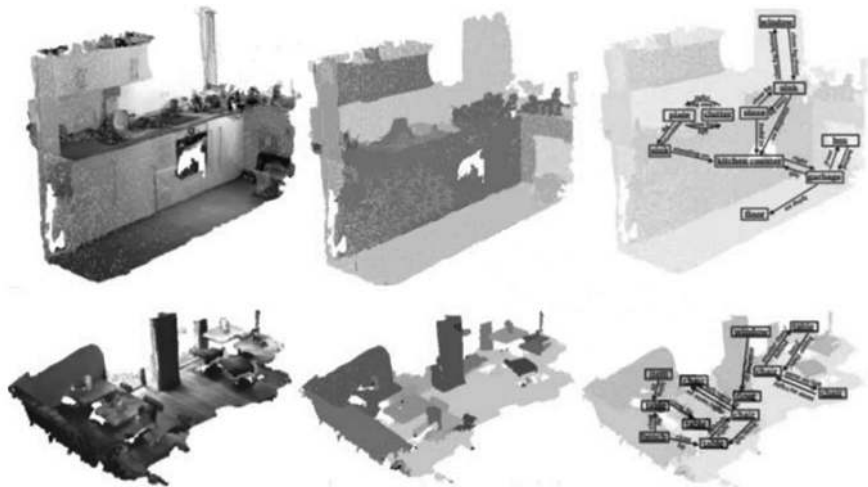


Fig. 3.12 Scene graph prediction results

The hierarchical object learning module uses coarse and fine-grained object labels to supervise the learning of the initial features of the object and the fine-grained object features after MP fusion. After being constrained by coarse-grained labels, the initial features of the object are fused with environmental context information and then constrained again using fine-grained labels. The hierarchical object learning module consists of two three-layer fully connected layers, which directly output object labels based on the input initial and fine-grained features of the object.

The fine-grained relationship features after MP fusion are used to predict relationship labels. The relationship predictor consists of three fully connected layers, which accept the fused relationship features and output relationship labels. The entity labels and relationship labels output by the hierarchical object learning module form the final scene graph on the graph skeleton. When the relationship label is predicted as “no” relationship, it means that there is no relationship between the two objects, i.e., the connection between the two points is deleted, and the final input 3D point cloud scene is represented as multiple scene graphs of different sizes.

3.3.2.4 Results Display

For the scene graph generation experiment, training was conducted on the 3RSCAN dataset, which has 1482 scene graphs. The experiment selected 160 frequently appearing object categories and 26 predicate labels in the scene as learning objects. By mapping the 160 object categories to NYU40, 40 coarse-grained object categories were obtained. As shown in Fig. 3.12, the method has greatly improved the prediction accuracy for original point clouds that are too incomplete or too small in quantity, but there are still cases of prediction failure. However, the method has

good prediction effects for the fine-grained labels of objects and the multi-angle prediction of relationships.

3.3.2.5 Summary

Due to the introduction of structural information, the problem of high redundancy and feature confusion caused by inappropriate visual features and indiscriminate fusion of environmental context in the task of 3D scene graph generation, which leads to difficulties in object and relationship prediction, has been effectively solved. By defining new visual features and interaction spaces, initial features with appropriate redundancy are extracted at the information extraction stage, and features with high discrimination are constructed by integrating structured skeleton information and hierarchical object granularity information, thereby extracting structured scene information.

3.4 Summary

The implementation process of the structured information of the scene graph can be seen as the process of organizing unstructured scene information into a graph using scene layout, i.e., spatial relationships. Therefore, the accuracy of the scene graph generation task is lower than that of the simple unstructured information extraction process; moreover, the graph structure of the scene graph is a form of inferable information organization, and the flat, unrelated scene information can be used for high-level scene knowledge inference after organization, which can help robots to achieve high-level tasks in the scene.

3.5 Discussion

- (1) Please explain what kind of high-level tasks structured and unstructured scene knowledge are, respectively, suitable for solving.
- (2) Please list the sub-tasks of the scene understanding task.
- (3) Please explain the advantages of using 3D point cloud data for scene understanding tasks compared to two-dimensional images.
- (4) Practice: Build a point cloud semantic segmentation and object classification network based on PointNet.
- (5) Practice: Try to build a scene graph generation network based on 3D scenes using PointNet.

References

1. QI C, YI L, SU H, et al. Pointnet++: deep hierarchical feature learning on point sets in a metric space[C]//Proceedings of the annual conference on neural information processing systems. LaJolla: NIPS, 2017(30): 5099–5108.
2. HU Q, YANG B, XIE L, et al. RandLA-Net: efficient semantic segmentation of large-scale point clouds[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2020: 11105–11114.
3. XU X, LEE G H. Weakly supervised semantic point cloud segmentation: Towards 10× Fewer Labels[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2020: 13703–13712.
4. JIANG L, ZHAO H, SHI S, et al. PointGroup: dual-set point grouping for 3D instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2020: 4866–4875.
5. ZHAO L, TAO W. Jsnet: joint instance and semantic segmentation of 3D point clouds[C]//Proceedings of the AAAI conference on artificial intelligence. Palo Alto, CA: AAAI, 2020: 12951–12958.
6. ZHANG H, DANA K, SHI J, et al. Context encoding for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2018: 7151–7160.
7. SHI W, RAJKUMAR R. Point-gnn: graph neural network for 3D object detection in a point cloud[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2020: 1711–1719.
8. SHI S, WANG X, LI H. Pointcnn: 3D object proposal generation and detection from point cloud[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Los Alamitos, CA: IEEE Computer Society, 2019: 770–779.
9. YU F, KOLTUN V. Multi-scale context aggregation by dilated convolutions[C]. International Conference on Learning Representations. San Juan, Puerto Rico: Conference Track Proceedings, 2016.
10. CHEN L C, PAPANDREOU G, KOKKINOS I, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence [J]. Los Alamitos, CA: IEEE Computer Society, 2017, 40(4): 834–848.
11. CHEN L, PAPANDREOU G, SCHROFF F, et al. Rethinking atrous convolution for semantic image segmentation[EB/OL]. (2017-06-17). <https://arxiv.org/pdf/1706.05587>.
12. LIN T, DOLLÁR P, GIRSHICK R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 2117–2125.
13. KRÄHENBÜHL P, KOLTUN V. Efficient inference in fully connected CRFs with gaussian edge potentials[C]//Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2011: 109–117.
14. YI L, SU H, GUO X, et al. Syncspecnn: Synchronized spectral cnn for 3D shape segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 2282–2290.
15. MASCI J, BOSCAINI D, BRONSTEIN M, et al. Geodesic convolutional neural networks on riemannian manifolds[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2015: 37–45.
16. BOSCAINI D, MASCI J, RODOIĆ E, et al. Learning shape correspondence with anisotropic convolutional neural networks[C]//Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2016: 109–117.
17. BOSCAINI D, MASCI J, RODOLÀ E, et al. Anisotropic diffusion descriptors. Computer Graphics Forum[J]. England: Wiley, 2016, 35(2): 431–441.

18. SINHA A, BAI J, RAMANI K. Deep learning 3D shape surfaces using geometry images[C]// Proceedings of the European conference on computer vision. Berlin, Germany: Springer, 2016: 223–240.
19. SIMONOVSKY M, KOMODAKIS N. Dynamic edge-conditioned filters in convolutional neural networks on graphs[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 3693–3702.
20. LI Y, PIRK S, SU H, et al. Fpnn: Field probing neural networks for 3D data[C]//Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2016: 307–315.
21. TULSIANI S, SU H, GUIBAS L J, et al. Learning shape abstractions by assembling volumetric primitives[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 2635–2643.
22. MARONA H, GALUN M, AIGERMAN N, et al. Convolutional neural networks on surfaces via seamless toric covers. ACM Transactions on Graphics. Association for Computing Machinery[J]. New York, NY: ACM, 2017, 36(4): 1–10.
23. QI C R, SU H, MO K, et al. Pointnet: deep learning on point sets for 3D classification and segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 652–660.
24. LANDRIEU L, SIMONOVSKY M. Large-scale point cloud semantic segmentation with superpoint graphs[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2018: 4558–4567.
25. GRAHAM B. Spatially-sparse convolutional neural networks[EB/OL]. (2014-09-22). <https://arxiv.org/pdf/1409.6070>.
26. BOSCAINI D, MASCI J, RODOLÀ E, et al. Learning shape correspondence with anisotropic convolutional neural networks[C]//Proceedings of the annual conference on neural information processing systems. La Jolla: NIPS, 2016: 3189–3197.
27. MONTI F, BOSCAINI D, MASCI J, et al. Geometric deep learning on graphs and manifolds using mixture model cnns[C]//IEEE/CVF Conference on Computer Vision and Pattern Recognition. Honolulu, HI, USA: IEEE Computer Society, 2017: 5115–5124.
28. QI X, LIAO R, JIA J, et al. 3D graph neural networks for rgb-d semantic segmentation[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 5199–5208.
29. BRUNA J, ZAREMBA W, SZLAMA A, et al. Spectral networks and locally connected networks on graphs[C]// International Conference on Learning Representations. Banff, AB, Canada: Conference Track Proceedings, 2014.
30. JOHNSON J, KRISHNA R, STARK M, et al. Image retrieval using scene graphs[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2015: 3668–3678.
31. XU D, ZHU Y, CHOY C, et al. Scene graph generation by iterative message passing[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2017: 3097–3106.
32. ZELLERS R, YATSKAR M, THOMSON S, et al. Neural Motifs: Scene graph parsing with global context[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. New York, NY: IEEE, 2018: 5831–5840.
33. YANG J, LU J, LEE S, et al. Graph R-CNN for scene graph generation [C] //Proceedings of the European conference on computer vision. Berlin, Germany: Springer, 2018: 690–706.

Chapter 4

Robot Navigation and Obstacle Avoidance



4.1 Overview

The task of visual localization [1–3] is to estimate the translation and rotation of the visual device through the acquired visual data. The essence of the task is to solve the transformation matrix of the data transformed between different coordinate systems. The visual localization task is actually a general task description, and the camera localization task is a subtask of it, aiming to solve the transformation matrix from the camera coordinate system to the world coordinate system. In actual operating environments, such as the situation where robots perceive scenes, camera localization tasks are usually classified according to actual application situations. When a robot enters a new scene, the primary task is to explore the environment extensively. During the entire exploration process, the robot operates without any prior knowledge of the environment, but the robot still needs to constantly perceive its own translation and rotation. In this situation, the camera localization task needs to be carried out while simultaneously exploring certain scene information. Such a camera localization task is called a visual odometry task. Visual odometry tasks usually need to temporarily save some information during the localization process. However, in real application environments, it is not necessary to constantly establish a large amount of temporary information during the localization process but to locate it with the help of known scene data. Such localization tasks are called re-localization tasks.

The visual re-localization task is to match the captured data with the known scene information in a known scene to obtain the accurate camera pose. There are many differences between the re-localization task and the visual odometry (VO) task: First, the main difference is that the re-localization task is based on a large amount of known scene data, without acquiring and storing temporary data. While the visual odometry task has no prior scene data at startup and relies less on global scene data during the entire localization process, the obtained camera pose is mainly a relative pose between frames, thus there is a more obvious cumulative error

problem. Second, the re-localization task usually only uses recently collected data. For color image (red-green-blue, RGB) cameras, it is usually one or more continuous images, and this is used to estimate the camera pose, and the dependence between these data during the localization process is weak. However, because the visual odometry task estimates the camera pose mainly based on relative poses, it not only depends on recently collected data but also depends on all scene data during the entire localization process to a certain extent, and the interdependence between these data is strong. The re-localization task has many application scenarios, such as in common scenarios of robot navigation in known scenes, and in augmented reality application scenarios, the re-localization task can be applied. The re-localization task can also assist the SLAM system in recovering the camera pose when the localization information is lost due to severe shaking or obvious light changes during the simultaneous localization and mapping [4] (SLAM) process, which can significantly improve the usability of the SLAM system. The loop detection task in the SLAM task is similar to re-localization, especially in some methodological treatments, but it is necessary to understand the difference between them. The purpose of the re-localization task is to obtain the pose, and the purpose of the loop detection task is to assist in correcting the pose of the SLAM system. These two tasks have essential differences in purpose, and they should not be regarded as the same task because of a certain similar method.

The navigation and obstacle avoidance of robots are based on map planning using established maps, and they rely on localization data to continuously move toward the destination. Navigation tasks and obstacle avoidance tasks are similar, but navigation tasks are usually considered to have planned routes, and robots advance according to the predetermined routes. This method not only requires a well-built model but also requires path planning based on the map, the set start and end positions, and continuous correction of the robot's behavior during its progress, eventually reaching the destination. The biggest difference between obstacle avoidance tasks and traditional navigation tasks is that they only understand the destination, and the complete route is not specially planned. This can cope well with real robot mobile scenarios. The robot only needs to continuously approach the destination during the movement and avoid obstacles around it during the progress. This method of robot navigation can adapt to more complex scenarios and is more in line with real application situations, such as indoor dynamic scenes with a large number of moving objects. Of course, no matter which localization method is used, it depends on the robot's estimation of its own position status information. Therefore, after introducing the related technologies of the relocalization algorithm based on known scenes in this chapter, we will focus on the related technologies of robot navigation and obstacle avoidance.

This chapter also provides reference codes for some technical cases mainly based on visual relocalization to help readers better understand and learn related technologies. These practical codes can provide you with some implementation ideas and details for reference. The specific codes for practice can be obtained by the SN-branded shortURLs below.

► sn.pub/ebn9id

4.2 Related Work

4.2.1 *Classic Relocalization Algorithm*

Knowing your exact location is key to many applications in the field of computer vision, such as adding a virtual thing to a scene shot on a smartphone, or navigating a self-driving car through a city. It can be found that these devices are usually equipped with RGB cameras because of their low cost and rich output.

SLAM or similar technologies are suitable for use when first entering a scene, but due to the design flaws, frame-by-frame pose calculation will lead to error accumulation, so loop detection and relocalization technologies are also involved in its application process. When SLAM loses its previous position due to factors such as blurring caused by high speed during operation, it needs to use relocalization technology, which is based on previous data, to recover the current approximate position. The accumulation of errors is corrected by loop detection, and the judgment of whether it is in the same position also needs to appropriately use relocalization technology.

Relocalization is localization with prior scene data, and its goal is to rely on prior known scene data to get the camera shooting pose of a query image. This localization technology saves a lot of costs compared to SLAM's instant and no prior localization knowledge, avoids error accumulation, and can solve the problem of position loss during the localization process. New methods like regression forest methods [5–8] and deep learning methods [1, 9–16] can avoid feature detection and matching, which are very resource-consuming calculations. These methods not only ensure that each localization is based on the same origin of world coordinate but also can seek pose from views not reached in prior knowledge. For the acquisition of prior knowledge of relocalization, SLAM technology is relatively mature, but its mechanism and defects make it suitable for obtaining unseen scenes, and for situations with prior knowledge, relocalization methods are more effective and convenient.

4.2.1.1 Geometric Methods

In recent years, with the development of deep learning technology, classic relocalization algorithms have continuously evolved into deep learning-based relocalization techniques. However, classic relocalization techniques still form the foundation of these technologies. Classic relocalization methods mainly include geometric methods and image retrieval methods. Geometric methods usually calculate feature data between images, form pairing relationships under different coordinate systems, and then solve for poses by solving triangles. Common feature extraction methods include oriented fast and rotated brief (ORB) [17], speeded-up robust features (SURF) [18], and scale-invariant feature transform (SIFT) [19]. However, these types of visual features usually have lower adaptability to scenes and larger

computational loads, resulting in poorer generalization performance across scenes, lower robustness, and lower operational efficiency.

Pose refers to position and attitude, corresponding to translation and rotation, respectively. Translation can usually be represented by distance in the Euclidean coordinate system, while rotation has multiple representation methods, such as rotation matrices, rotation vectors, quaternions, and Euler angles. This section mainly uses rotation matrices for representation due to their computational simplicity and relative ease of conversion with other representations.

As outlined, the goal of the camera relocalization task is to estimate the camera pose in a known scene. The camera pose is the linear transformation relationship between the camera coordinate system and the world coordinate system. The relationship between coordinate systems can be referred to in Fig. 4.1. Therefore, the solution of the camera pose can actually be transformed into describing the same spatial point in the same world coordinate system, under different positions and angle forms, and also being able to convert between different coordinate systems.

The solution of the camera pose is to try to find the coordinate transformation relationship between the points observed under the current camera coordinates and the points in the world coordinate system. Through the representation of the pose, it can be known that translation and rotation each require at least three variables to be

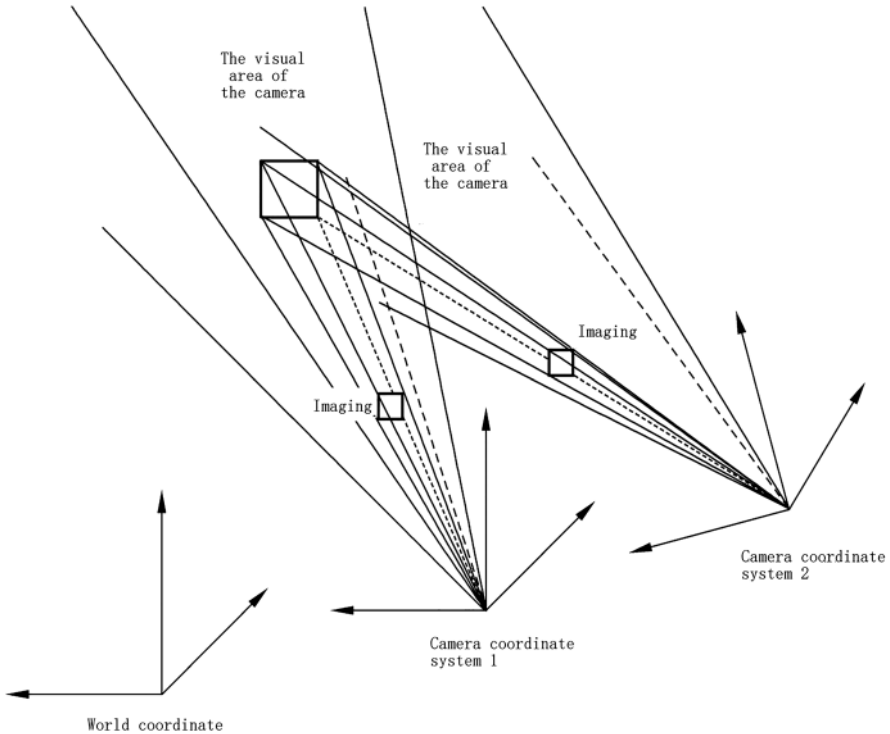


Fig. 4.1 Relationship between camera and world coordinate systems

solved, so at least three equations need to be established. The key to solving the transformation relationship is to establish equations. The basis for constructing equations is to describe the same thing in different ways. In the same scene, a certain point between two coordinate systems is converted to each other, which is the basis for constructing equations. Therefore, how to determine the method of the same point expressed in these two coordinate systems is the key to solving the camera pose. The point in the world coordinate system is usually called the scene point, and the point on the image taken by the camera is called the pixel point. When they describe the same point, they are called corresponding points.

Therefore, how to establish the corresponding relationship between the world coordinate system and the camera coordinate system becomes the key issue in solving the camera pose. Usually, the determination of the world coordinate system is determined by the camera coordinate system at the starting location of the first camera. At the same time, in order to ensure the uniformity of feature description, the establishment of corresponding points is best done under the same data expression form. The most common is to extract feature points on the image.

Matching feature points in the image is a key step in obtaining the corresponding relationship. In the work of extracting feature points from images and matching them, some researchers have proposed some effective methods, such as the famous ORB, SIFT, and SURF. These artificially designed feature points have some very good properties, including repeatability, distinguishability, high efficiency, and locality. Feature points are composed of keypoints and descriptors.

In the algorithm for feature extraction, the SIFT feature point extraction algorithm is very effective. It fully considers the problems that may be encountered in matching, such as illumination, scale, and rotation. However, its biggest drawback is that it requires a large amount of computation, and the current CPUs are still unable to run the SIFT feature point extraction algorithm in real time. Not only SIFT but SURF and ORB also have the same problem. Among them, ORB is relatively faster, but it still takes about 20 ms to process a frame.

Below is a simple description of the steps of the SIFT feature extraction algorithm:

- Step 1: Input the image and convert the image to a grayscale image.
- Step 2: Process the grayscale image to obtain an image pyramid, which is generally a Gaussian pyramid.
- Step 3: Convolve the image with a Gaussian template to obtain a Gaussian difference pyramid.
- Step 4: Perform preliminary detection of the extreme points in the Gaussian difference pyramid and locate the key points.
- Step 5: Collect the pixel gradient and direction distribution in the neighborhood window of the Gaussian pyramid for the collected key points. Determine the main direction of the feature points.
- Step 6: Generate the descriptor of the feature point. This part involves determining the area required for calculating the descriptor, rotating the coordinate axis to the main direction, generating the gradient histogram, trilinear interpolation, and descriptor thresholding operations.

After obtaining the feature points, it is necessary to match the feature points. Although the ideas for matching may vary, they all hope to find the point with the smallest distance between the two matching points. Most third-party libraries use Euclidean distance or the angle of the vector to obtain matches. When searching for matches, K-nearest neighbor (KNN) and K-dimensional tree (KD tree) methods are more commonly used.

The research on SLAM has realized a localization method without prior scene knowledge. This method continuously performs feature matching between video frames to obtain the motion trajectory of the camera. However, this type of method has great limitations, especially when re-entering a scene that has been visited before. SLAM not only needs to consider the accuracy of the pose information of the previous frame when localization between frames but also needs to consider whether the depth information of the pixel point position is accurate. The errors caused by these uncertain factors accumulate over time, causing the pose information of the camera to become increasingly inaccurate and a significant offset occurs. In addition, the camera is easily lost due to the two characteristics of high speed and weak texture during the localization process. In order to solve these problems, techniques such as loop detection have been proposed, which is the precursor of relocalization. It tries to find previously visited locations to correct the pose of the camera, and this involves the problem of camera relocalization.

Camera relocalization based on RGB has received a lot of attention in recent years. After collecting data for some scenes, when re-entering the scene, the position and posture of the camera can be directly obtained. This is a key support for many existing technologies. The camera pose obtained by relocalization can not only meet this requirement but usually can also obtain a relatively stable pose during the localization process according to previous experience. The accuracy of the current camera relocalization has been greatly improved, and the following is a detailed introduction to the classic camera relocalization methods.

Random Sample Consensus Algorithm Finding a function that fits the distribution of a set of data is a relatively difficult problem. When looking for a suitable function, the relationship between them must be considered. The least squares method is a good way to do this. The least squares method is a basic data optimization method that can find the desired fitting function by minimizing the square of the error. This method only works well when there are few outliers. When the proportion of outliers is large, it cannot produce good results. The outliers mentioned here can be understood as values that exceed the reasonable fluctuation range of the expected fitting function. The corresponding inliers are points within the reasonable fluctuation range.

Because the least squares method cannot handle too many outliers when fitting functions, the random sample consensus algorithm [20] (RANSAC) was born. This algorithm fits some data randomly selected, then finds the inliers of these fitted functions again, and chooses the best fitting function based on the number of inliers. The specific steps of the algorithm are as follows:

- Step one: Randomly select a subset (usually less than 1/2) A from the full set U as the data to be fitted.
- Step two: Fit the selected subset data A to get the first fitting function f_1 .
- Step three: Repeat the first two steps until the number of fitting functions generated by sampling reaches the set number k . At this time, k fitting functions $f_1, f_2, \dots, f_k$ are obtained.
- Step four: Set a threshold T to distinguish inliers from outliers, and calculate the number of inliers for k fitting functions $f_1, f_2, \dots, f_k$ on all points in the full set U .
- Step five: Select the fitting function f_m with the most inliers, fit it with all its inliers, and get the first optimized function f'_1 .
- Step six: Calculate the inliers of f'_1 on the full set U again.
- Step seven: Update f'_k again based on the obtained inliers.
- Step eight: Repeat steps six to seven until $f'_k - f'_{k-1} < T'$, where T' refers to the difference between the two functions. This difference can be defined by yourself, for example, when the parameter difference between the functions is very small, you can stop optimizing, or stop optimizing when the number of inliers obtained by the two functions is the same, etc.

Figure 4.2 provides a visual description for intuitively understanding the RANSAC algorithm. In Fig. 4.2a, some points are randomly selected and some fitting functions are generated. In Fig. 4.2b, the fitting curve with the most inliers is selected and its inliers are used for further fitting. In Fig. 4.2c, after obtaining a new fitting curve, its inlier set is obtained again and fitted. This process is repeated until the number of inliers remains stable, and the final result is obtained. It can also be seen from the process that although there are many outliers to be fitted, setting a threshold can effectively suppress outliers and obtain as good a function as possible. Therefore, setting a threshold is a very critical step in the RANSAC algorithm. It is not appropriate to set it too large or too small. A suitable threshold can make the function fit the function as well as possible.

4.2.1.2 Image Retrieval-Based Methods

Image retrieval-based methods are also classic camera relocalization schemes. This method usually extracts features from collected reference images. Common feature extraction includes vector of locally aggregated descriptor (VLAD) features [21], and in recent years, there has been the emergence of networked local aggregation descriptor vector features (NetVLAD) [22]. The extracted features will be used to build an image-level feature database. When an image that needs to be located is input, the entire image is subjected to feature extraction, and the extracted features are compared with the features in the image feature database to obtain its approximate pose. If you want to improve the accuracy of the location, you can extract and match feature points based on the query image, and then use the random sampling

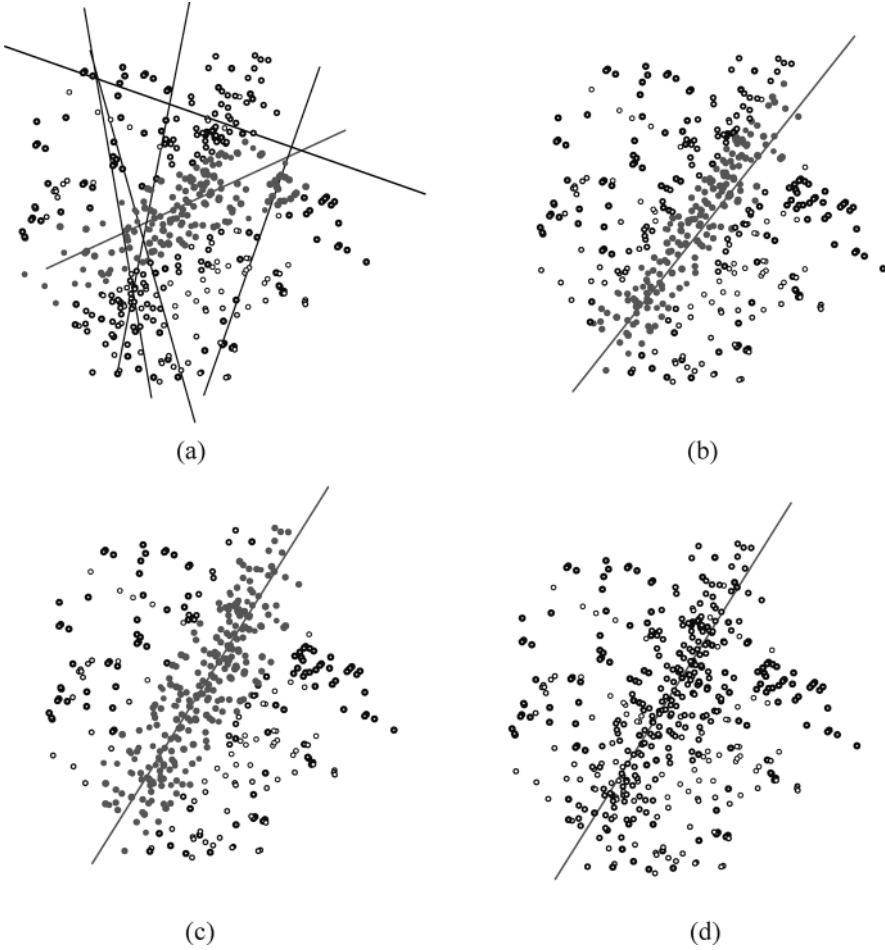


Fig. 4.2 RANSAC optimization process

consistency method and the multi-point perspective algorithm (perspective-n-point, PnP) to further solve the more precise camera pose.

In the SLAM problem, the most commonly used solution for loop detection is achieved through image retrieval. Although this method has certain limitations, it received a lot of attention and made some good progress before the popularity of machine learning and deep learning methods. In the loop detection problem, the purpose is to correct the deviated pose, because in the SLAM method, the camera's motion tracking is done frame by frame, and as the error of each frame continuously propagates to the next frame, the camera's motion keeps deviating more and more. The longer the time, the greater the deviation. In addition, the method of obtaining image matching between images based on feature detection has certain requirements for the imaging quality and content of the image, so the camera may also lose

its pose due to too fast motion and lack of texture in the scene during motion tracking.

The relocalization technology based on image retrieval is achieved by establishing an image database with geographical locations. The successful implementation of this method of establishing a large database is due to the development of image retrieval technology. The features of the image are usually described with VLAD descriptors. The task of loop detection is to judge whether the camera has previously arrived near the current location of the camera. If it has a similar pose, it must have taken a similar picture. In view of this, whether it is from the perspective of loop detection or from the perspective of image-based relocalization, finding similar images is a feasible solution to obtain the camera pose.

In order to find similar images, relocalization based on image retrieval has adopted different types of image representation and retrieval methods. The simplest and most effective way is to extract the VLAD features of the entire image in the database and store these features. When an image that needs to be searched is input, the query image is extracted with VLAD features and matched with the features of all database images. The one with the highest match score is the most similar and the closest pose image. However, the computational cost of this method is too high, the accuracy of camera localization is relatively low, and it is almost impossible to achieve real-time localization. At the same time, as the image database grows, the cost of matching images will become larger and larger, so it is not applicable in most real-time systems.

Another method based on image similarity is the most well-known based on the bag-of-words model. The bag-of-words model means that each image has a “bag” of “vocabulary” that describes itself. By counting the number of “vocabulary” that each image has in common, it can be judged whether the images are similar. All the vocabulary in the bag-of-words model constitutes a dictionary. The method of generating a dictionary uses the K-means algorithm [23] (an iterative solution clustering analysis algorithm). The process is shown in Fig. 4.3. If you want to generate a dictionary with n words, first drop randomly, each feature chooses the nearest vocabulary center to itself as its own type, then the vocabulary center moves according to the features that choose itself, and then each feature chooses the nearest vocabulary center to itself as its own type again, and the vocabulary center moves again. This iteration continues until the vocabulary center tends to be stable. Figure 4.4 gives an intuitive visualization process.

For the matching method of the bag-of-words model, selecting matching words aimlessly will waste a lot of computing resources, so it is necessary to consider using weighted features. The idea of term frequency-inverse document frequency (TF-IDF) [24, 25] is very effective. For an image, the words with high frequency are high-discrimination words, and the words with lower frequency in the overall dictionary are low-discrimination words. Selectively matching words makes the efficiency higher.

Looking up the dictionary requires excellent data representation to quickly find similar images for each image. Compared to searching one by one, building a KD-Tree or K-ary tree [26] is a very good search method.

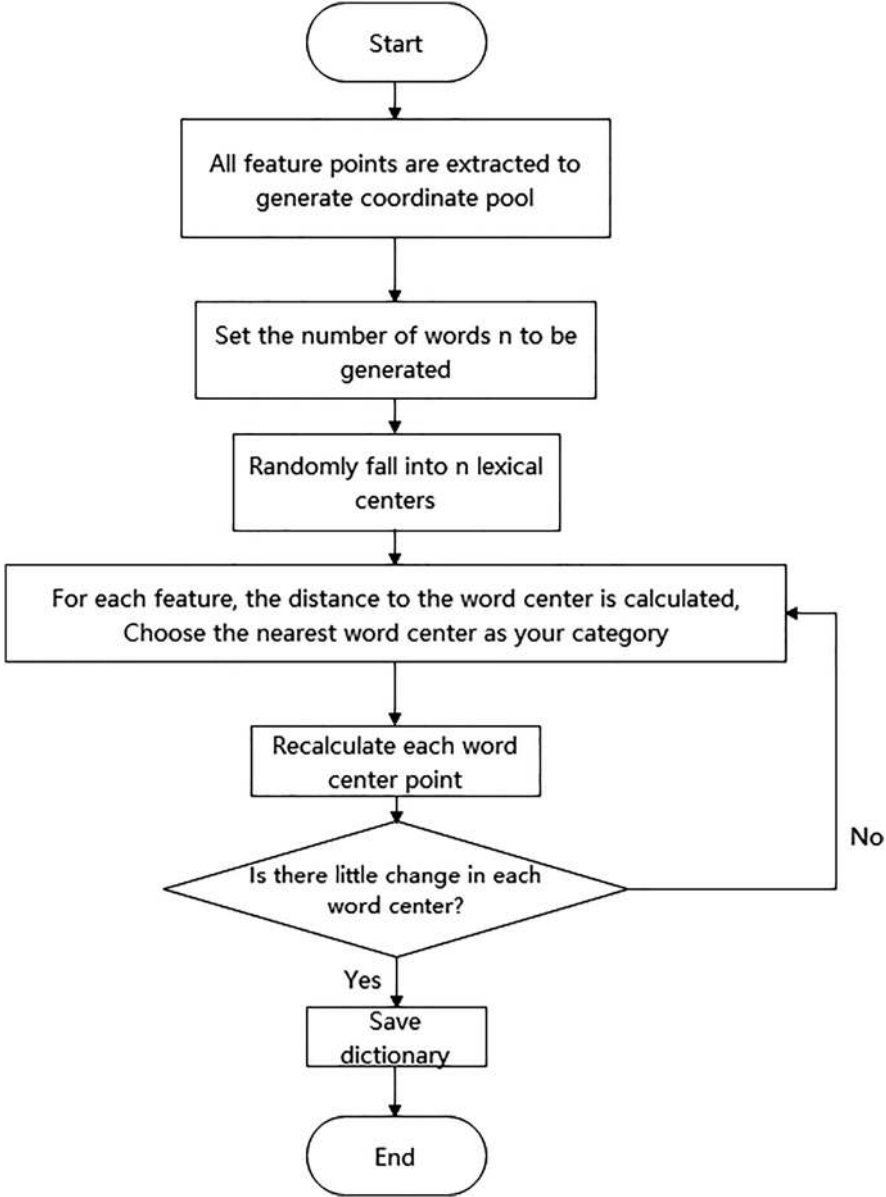


Fig. 4.3 Flowchart of dictionary generation in the bag-of-words model

With the development of deep learning technology, image feature extraction can be done not by using SIFT, SURF, or ORB feature extraction algorithms, but directly through pre-trained deep neural networks for feature extraction [27] as its features, then using the DenseSIFT [28] algorithm for matching, and finally using virtual

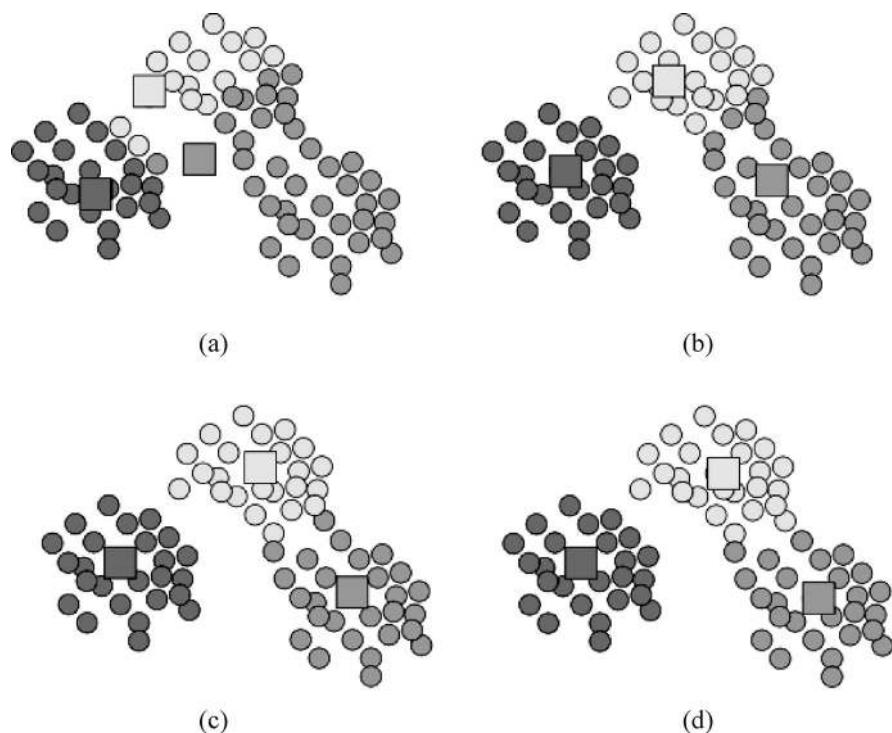


Fig. 4.4 Visualization process of vocabulary center iteration. (a) Randomly drop three vocabulary centers; (b) The vocabulary center adjusts its position to the center of the feature according to its own selected features; (c) Continuously repeat and adjust the vocabulary center; (d) Generate the final dictionary when the vocabulary center changes slightly

view synthesis verification to determine the final camera pose. The pose generated by this method is quite accurate and worth considering in image retrieval-based methods, but this method is less efficient and the scale of the database is usually large, making it inconvenient to use.

4.2.1.3 Relocalization Technology Based on Random Forests

Compared to image retrieval methods, using random forests to solve the relocalization problem can be more effective. Shotton et al. [5] designed a method for camera relocalization using random forests. This method predicts the scene coordinates corresponding to 2D points and finally estimates the camera pose of the image in a RANSAC process. This method can be said to have pioneered excellent pose results in machine learning for camera relocation. In subsequent methods, the efficiency and accuracy of random forests are constantly improving, but random forest methods require the addition of depth information. Even if subsequent work can avoid

using depth information in practical applications, depth information is still necessary during training.

Applying the scene coordinate regression idea of random forests to deep learning can also estimate the scene, and specially designed loss functions can even avoid using depth information. There are even techniques based on deep learning that can directly regress the camera pose.

4.2.2 Deep Learning-Based Relocalization Algorithm

Relocalization algorithms based on deep learning can be roughly divided into two strategies: one is to directly obtain the pose, such as directly regressing the camera's pose [9] or matching in frames with known poses [29–31]; the other is to try to establish a correspondence between points on the image and their world coordinate points, and then use the PnP algorithm to generate the best pose estimate in a RANSAC algorithm [20] process.

4.2.2.1 Directly Regressing the Pose

In the methods of directly regressing pose, one category is based on image (frame) localization technology, which benefits from the development of image retrieval technology [22, 32, 33]. These methods query matching images from a database of images (frames) with pose information and then calculate the pose of the query image through the matched image with pose. These methods can be applied in large-scale scenes, but they usually can only provide rough estimates of camera pose. Accurate 6 degrees of freedom (DoF) camera pose needs to be obtained using sparse feature-based methods. Sparse feature-based methods match image descriptions with 3D points reconstructed after motion, obtaining a series of 2D–3D correspondences, from which accurate pose estimation information is obtained [34]. In these methods, the focus of research is on how to improve the efficiency of description matching [35], robustness [35, 36], and the ability to extend to large-scale outdoor scenes [37–39]. However, local feature detectors rely on rich textures and good imaging quality [20], which makes the reconstruction of scenes with a large number of repeated elements very difficult. Compared to the matching database, Kendall et al. [9] proposed the PoseNet algorithm, which is a method of directly regressing pose. With the help of transfer learning technology, it directly maps an image to a 6 DoF camera pose through a CNN network. Because the PoseNet algorithm treats the method of camera pose estimation as a regression problem and directly uses neural networks to solve this problem, it does not require traditional feature extraction. This method improves accuracy by using reprojection loss and more expressive buildings and can succeed in challenging scenes where traditional methods fail. However, the accuracy of this method is very low, and although some subsequent

schemes can improve the pose estimation effect to some extent, their accuracy improvement is limited.

4.2.2.2 Indirect Solution of Camera Pose

Compared to directly regressing pose, another solution is to decompose the localization method into several steps, and the steps suitable for using machine learning are then implemented using machine learning algorithms. This is also a commonly adopted method recently, and it usually produces better pose calculation results. Recently, Shotton et al. [5] learned to predict 3D scene points of image blocks through a random forest and used these corresponding points to obtain the camera's pose, which improved the accuracy of subsequent work [6, 40], and realized a fast learning method to obtain 3D scene coordinates [8]. However, these methods are very dependent on depth information, and although subsequent algorithms [7, 41] use the method of random forest mapping, these methods still need to use depth maps or 3D models for training. However, obtaining depth maps is not always easy, so this method is more friendly for indoor scenes but somewhat powerless for larger outdoor scenes. Brachmann et al. [3] proposed a differentiable RANSAC algorithm process and used it in end-to-end network training for camera pose acquisition, naming it the DSAC algorithm (differentiable RANSAC, DSAC). It mainly uses a visual geometry group from the University of Oxford.

The convolutional neural network (CNN) in the style of the Visual Geometry Group (VGG) maps image blocks to corresponding points in the scene space, i.e., obtaining scene coordinates. Then, a scoring CNN is used to calculate the scores of the poses obtained from the random four scene coordinates calculated by the P3P algorithm. Finally, the selected pose is optimized through a weighted RANSAC mechanism to obtain the final pose. The full-frame network method was proposed in literature [42]. This network uses an encoder and decoder network to obtain the same number of scene coordinates as the size of the input image, which is different from the way the DSAC algorithm predicts image blocks. Considering that global information is very prone to overfitting, a large amount of variant data is generated using the training set data to increase the size of the dataset to avoid overfitting, which also improves the performance to some extent.

The DSAC++ algorithm [1] is an upgraded version of the DSAC algorithm. Compared to the DSAC algorithm, it has made optimizations in many aspects. First, it uses a fully convolutional network to more efficiently generate more scene coordinates, replacing the differentiable RANSAC with softened inlier statistics, and when choosing the optimal pose, it uses a flexible maximum distribution probability method. Empirical evidence shows that this method is more robust than training a CNN scoring network. The greatest contribution of the DSAC++ algorithm is that it proves that scene coordinate regression can be learned even without depth information. However, for regression forest methods, whether depth information is used in the application process or not, depth information needs to be used for training,

because depth information provides a large amount of scene structure information, which can greatly reduce the difficulty of learning the regression forest.

The DSAC++ algorithm divides the entire learning process into three parts. The first part is the initialization part, which uses the Euclidean distance between the estimated value and the real scene coordinate value as the objective function to train the network, so that all predicted results can initialize the network's predicted scene coordinates as much as possible in front of the camera. If this training step is missing, the subsequent training will not converge. When not using depth information, this training part uses a heuristic method to generate fake depth as initialization. The second part further trains the network through reprojection error, which is very important, especially when the first step of training does not use depth information, this part of the training will effectively improve the performance of the network. The third part performs end-to-end optimization steps to complete all network training. Recently, an article [43] redefined the second part of the DSAC++ algorithm's loss, proposing an angle-based reprojection loss. This loss function overcomes the problem of careful initialization, combining the first two steps of DSAC++ algorithm training into one step, saving a lot of training time, and allowing multi-view constraints, which also improves the performance to some extent.

In summary, the two most competitive methods currently are the random forest-based method and the deep learning-based method. The deep learning-based method can now be trained without using depth information, while the random forest-based method cannot be trained without using depth data. Considering that the acquisition of depth information itself is quite difficult, and the depth camera has great limitations that prevent it from effectively extending to outdoor scenes, the deep learning method has greater advantages in terms of training data requirements, which also allows it to be applied in more scenes. This book is also based on this consideration, to further research on deep learning-based methods.

4.2.3 *Classic Navigation and Obstacle Avoidance Algorithms*

In the early stages of research, traditional robot obstacle avoidance methods could safely and effectively implement mobile robot obstacle avoidance in simple scenes. However, the inherent shortcomings of traditional obstacle avoidance methods make them difficult to apply in most scenes, for example: Traditional methods require a lot of time to map the environment with high precision. The huge amount of parameters is difficult to adjust during scene migration. In dynamically complex scenarios, localization may fail. Moreover, traditional methods mostly adopt centralized control methods, that is, all robots in the scene are uniformly dispatched by the central control system. The reason why traditional methods use a central control system is that it needs to make the correct actions based on the position and distance information of other mobile robots around and its current state, such as the optimal reciprocal collision avoidance method (ORCA). The problem to be solved by this method is that n ($n > 0$) mobile robots can dynamically avoid other mobile robots or

(dynamic or static) obstacles during the navigation to the target point and find the best path to move to the target point safely and efficiently. However, the dynamic obstacle avoidance algorithm based on ORCA has strict requirements. Assuming that there are n mobile robots in the scene, n mobile robots use the same collision avoidance strategy to avoid each other. Under sufficient external environmental conditions, the n mobile robots in the scene need to plan their own collision avoidance schemes within time t and safely reach the target location. The ORCA method plans the obstacle avoidance scheme according to the speed of each mobile robot, which means that each mobile robot in the scene needs to obtain the speed of other mobile robots to avoid collisions with other mobile robots in the scene. For each other robot in the scene, the current robot has a half-plane (velocity space) velocity selection space to choose its own velocity, thereby avoiding dynamic collisions with other mobile robots.

However, the computational load of the centralized control system increases exponentially with the increase in the number of mobile robots in the scene, so the centralized control method is difficult to deploy in large-scale application scenarios; moreover, in order to achieve safe and effective movement of mobile robots, the central control system needs to frequently and without delay send control instructions to mobile robots. This requires that the mobile robots in the system can frequently perform accurate localization, the central control system has super-strong computing power, and the speed of network transmission is fast enough. But these requirements are difficult to achieve in real scenarios, so the traditional method has not solved the problem of mobile robot obstacle avoidance very well.

A* Algorithm (A-Star Algorithm) The core of this algorithm is the estimation function $F = G + H$, where G is the actual distance moved from the initial position to the current position, and H is the estimated distance from the current position to the destination. The basic idea of the A* algorithm is: Select the point with the smallest F value from the open table, delete it from the open table, and add it to the close table, add the adjacent nodes of this node to the open table, check whether there is an endpoint, if there is, end the search and record the path. If there is no endpoint, repeat the previous process until the endpoint is reached.

The flowchart of the A* algorithm is shown in Fig. 4.5.

4.2.4 *Deep Learning-Based Navigation and Obstacle Avoidance Algorithm*

In recent years, with the rapid development of deep learning, more and more researchers have begun to try to use deep learning to implement learning-based mobile robot navigation methods. Pfeiffer et al. [44] proposed a data-driven end-to-end control method. They used the navigation package provided in ROS to collect a large amount of data in simulated scenarios and used the collected data to train deep

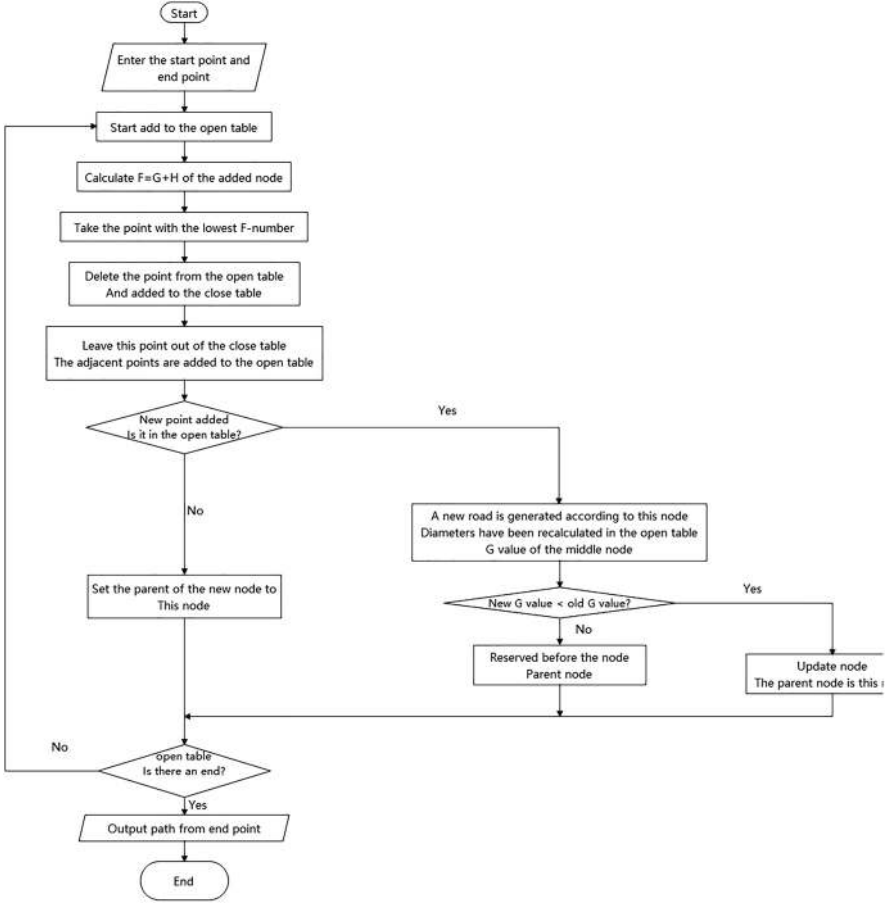


Fig. 4.5 A* algorithm flow

learning models. Based on the powerful feature representation ability of deep learning, training deep learning models in a data-driven manner can achieve effective mapping from complex laser data, target points, and velocity at the previous moment to simple behavior data. The trained deep learning model can achieve efficient obstacle avoidance of mobile robots in unknown scenarios and can help mobile robots cope with sudden situations in the surrounding environment. Like other supervised deep learning models, the obstacle avoidance performance of the obstacle avoidance algorithm based on the deep learning model heavily depends on the quality of the dataset.

To overcome the limitations brought by data, researchers have begun to try to use reinforcement learning to solve obstacle avoidance problems. Reinforcement learning, also known as enhanced learning, evaluative learning, and reinvigorating learning, is currently an important paradigm and methodology in machine learning.

Reinforcement learning guides the intelligent agent to continuously interact with the environment to collect experience data, and guides its own learning through the collected experience data, so that the intelligent agent can obtain the maximum cumulative return or achieve a predetermined goal in the interaction with the environment. Therefore, the characteristics of reinforcement learning make it more suitable for decision-making tasks such as mobile robot navigation. Deep reinforcement learning combines the powerful representation ability of deep learning and the decision-making ability of reinforcement learning, making it more suitable for the navigation tasks of mobile robots. In this field, Tai et al. [45] first proposed a map-less motion model based on deep reinforcement learning. The model maps high-dimensional observation data to behavior, enabling mobile robots to navigate safely in a priori unknown environments without collisions. However, due to the sparsity of laser data and the simplicity of the training scene, the model does not perform well on dynamic obstacles. To solve the above problems, Long et al. [46] proposed a multi-scene, multi-stage training framework based on the proximal policy optimization (PPO) deep reinforcement learning algorithm. By inputting continuous three-frame laser data, the mobile intelligent agent can successfully complete efficient obstacle avoidance in dynamic scenes. Based on this, Choi et al. [47] proposed an actor-critic deep reinforcement learning framework that introduces the LSTM method to reduce the limitations of sensors in the field of view and successfully used 90°FOV (field-of-view, sensor coverage angle) laser data to achieve the effect of 180°FOV laser data.

However, laser data is only one-dimensional distance information, so it is often difficult to accurately measure the distance from irregular objects to the intelligent agent, causing the intelligent agent to fail to avoid obstacles; the high price of laser sensors makes it difficult to use on a large scale in real life; lasers do not reflect when encountering special materials, resulting in low robustness to the environment. To solve the problems of lasers, a large amount of work has been done to use RGB data to solve the navigation and obstacle avoidance problems of mobile robots, such as optical flow prediction methods [48], vanishing point detection methods [49], visual SLAM (Visual SLAM, VSLAM) methods [27] and end-to-end deep learning methods [50]. Tai et al. [45] proposed an end-to-end obstacle avoidance model, achieving a mapping from depth maps to behavior. However, it is inconvenient and very time-consuming to control mobile robots to collect data in real scenes, so Ma et al. [51] and Bharadhwaj et al. [52] trained a reinforcement learning model that can achieve a mapping from RGB data to behavior in a simulated environment, and achieved good results in the simulated environment. Due to the huge differences in texture, color, etc. between the simulated environment and the real environment, the obstacle avoidance model trained in the simulation is difficult to transfer to the real environment. Gordon et al. [53] used the highly realistic Habitat [54] simulation environment to train visual navigation models, reducing the gap between simulation and reality, but it needs to retrain the model when transferring to new scenes. Chen et al. [55] proposed a standard for evaluating the gap between simulation and reality, and on this basis proposed a visual obstacle avoidance

method, but due to the use of imitation learning relying on big data and a simple training scheme, this method is difficult to apply to challenging tasks.

4.2.5 Summary

Navigation and obstacle avoidance is a very important task in robot applications. The basic tasks include camera localization, camera relocalization, path planning, and navigation obstacle avoidance. This section introduces the main related work on the topics of camera relocalization and navigation obstacle avoidance. In practical applications, most of the time, the application scenario already has certain scene prior information, which makes camera relocalization play an important role. Therefore, this section provides a detailed introduction to the task of camera relocalization. In the classical camera relocalization methods, the classical geometric methods are introduced, and some basic algorithms of this method are introduced, including the SIFT feature points used for feature point extraction, and the random sampling consensus algorithm used for pose solving. For classical methods, methods based on image retrieval and their basic algorithms, including the bag-of-words model and its process, are also introduced. Finally, the relocalization technology based on random forests in classical camera relocalization methods is introduced. Deep learning is a very popular technology in recent years, and it has many differences from traditional classical methods. Therefore, this section also introduces the camera relocalization methods based on deep learning, classifying them into direct pose estimation methods and indirect pose estimation methods, and introduces these algorithm technologies. In navigation and obstacle avoidance algorithms, the core path planning algorithm of navigation—A* algorithm is introduced. At the same time, navigation and obstacle avoidance algorithms based on deep learning are introduced, including the very important part of the navigation algorithm, the observable Markov decision process.

In order to understand the camera relocalization algorithm and navigation obstacle avoidance algorithm in more detail, the later chapters provide specific practices for the algorithms introduced earlier. The task of camera relocalization is implemented on the 7Scenes dataset provided by Microsoft. For classical camera relocalization, camera relocalization based on feature database matching is carried out, not only introducing its basic principles and knowledge but also providing pseudocode and specific implementation code, which can be accessed through the link for understanding. For camera relocalization based on machine learning, experiments are conducted using a network model combined with DenseNet and pyramid pooling. The practice part also introduces some specific knowledge needed for experiments under this model, and also provides code. For the navigation task, robot navigation practice based on camera relocalization is carried out, introducing ROS and map-building schemes, and introducing the specific process of path planning. Since the code implementation of this technical case is based on ROS, the code is simple, and everyone can easily practice it after

explaining the principle in this section. For the classic obstacle avoidance algorithm in the obstacle avoidance task, robot obstacle avoidance practice based on known grid maps is carried out, introducing navigation map construction methods and obstacle avoidance technical methods. For deep learning obstacle avoidance, a multi-agent obstacle avoidance practice based on deep reinforcement learning is designed. This practice uses ROS-distributed algorithms to implement multi-intelligence obstacle avoidance and uses simulation methods for experiments, which allows everyone to get started quickly. In practice, the path planning based on the DQN algorithm and the training of the DQN algorithm are introduced. The practice code of this part is also relatively simple, and interested readers can implement it themselves.

4.3 Technology Practice

4.3.1 Case Study: Camera Relocalization Based on Feature Database Matching

4.3.1.1 Task Description

In order to understand traditional camera relocalization schemes more clearly, this section introduces an implementation case of a classic relocalization method, a camera relocalization method based on feature database matching. The method based on feature database matching can be based on full-image feature matching or local pixel matching. Since the accuracy performance of the database matching method based on full-image feature matching is poor, this technical case part adopts the camera relocalization based on feature database matching as the localization case. The main content involved in the camera relocalization method based on feature database matching includes extracting feature point sets from images, storing feature point sets in a database, extracting features and querying features of retrieved images, and solving the pose of the matched feature relationship. The following will introduce these related methods and show the implementation details. The code for this technical case part is all based on Python implementation.



Chapter Knowledge Points

ORB feature extraction algorithm: A feature extraction algorithm proposed in 2011, as a fast alternative to SIFT and SURF feature detection and extraction algorithms, has stronger practicality.

K-means algorithm: A typical clustering algorithm, the representative interpretation model is the pastor-villager model, an algorithm that initializes based on randomly initial K points, each sample calculates its distance to the

K cluster centers and assigns it to the class corresponding to the cluster center with the smallest distance, and iterates until the termination condition.

Feature matching: A method of matching the similarity of two features after obtaining features, the most common matching methods are the sum of absolute differences (SAD) and the sum of squared differences (SSD).

Random Sample Consensus Algorithm: It is an iterative optimization algorithm used to deal with the method when there are a large number of outliers in the sample. This method first samples a large number of candidate results, screens candidate points according to the discrimination index, and optimizes the final result based on this.

4.3.1.2 Related Work

Extraction of ORB Features

ORB is a fast feature point extraction and description algorithm. This algorithm was proposed by Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski in a 2011 article titled “ORB: An Efficient Alternative to SIFT or SURF.” The ORB algorithm is divided into two parts, feature point extraction and feature point description. Feature point extraction is developed from the FAST (features from accelerated segment test) algorithm, and feature point description is improved based on the BRIEF (binary robust independent elementary features) feature point description algorithm. ORB features combine the detection method of FAST feature points with the BRIEF feature descriptor and have made improvements and optimizations on their original basis. The speed of the ORB algorithm can reach 100 times that of SIFT features and 10 times that of SURF features. Considering the practicality of ORB features, this technical case considers using ORB as the main method of feature extraction. Its extraction code can be implemented by calling the OpenCV library.



Sample code

```
import cv2

import numpy as np

img1 = cv2.imread("data/face1.jpg",0)# Import grayscale image

detector = cv2.ORB_create()

Kp1 = detector.detect(img1, None)

kp1 = detector.detect(img1, None)
```

Construction of Feature Database

Since the number of extracted features is usually very large, in order to efficiently perform matching and feature classification, the extracted feature points can be clustered to form a feature package corresponding to a certain class of similar features. In this technical case, the classic K -means algorithm is used for implementation. The K -means algorithm simply and effectively generates K classes from N data, which is a typical clustering algorithm, and has been introduced in the previous text. Here it is described in pseudocode, and the specific code can be accessed and downloaded from the SN-branded ShortURLs provided in this section.



Sample code

```
K_centers = random.(x,y)

While(True):

    Class = allPoints(x)

    For Pi in allPoints:

        Distances = Pi-K_centers

        NewClass(Pi) = Min(Distances)

    If NewClass-Class<Threshold:

        Break

    Else:

        Class=NewClass
```

Matching of Feature Point Sets

We have already extracted the feature points of the image and built a K -means database based on these feature points. The next step is the most critical part of the camera relocalization pose-solving process, which is to match the feature point sets. The features of the input image will be extracted and input into the created database for retrieval, so as to obtain the scene coordinates corresponding to its features. In this way, the relationship between the features and the scene is established. With the help of this relationship, the equation can be established and solved, and the pose of the camera can be estimated. The following pseudocode explains the process of matching feature point sets. For the specific code, please visit the SN-branded ShortURLs in the overview section of this chapter.



Sample code

```

InputPic = readImage(Path)

Features = cv2.ORBDetector(InputPic)

For feature in Features:

    Mindistance = threshold

        For class in AllClass:

            If class-feature<threshold

                Mindistance = abs(class-featuen)

                Fclass = class

```

4.3.1.3 Technical Method

Dataset

After designing the objective function of the scene coordinate network model and the weights used, consider selecting a specific dataset for training. In this technical case, we use the public dataset seven Scenes Dataset published by Microsoft Labs [5], which is also a dataset commonly used in recent localization-related research. This dataset contains seven indoor scenes, namely, Chess, Fire, Heads, Office, Pumpkin, RedKitchen, and Stairs. Figure 4.6 shows the 3D model effect and motion trajectory of each scene. The amount of data for each scene varies, and the specific quantity information is shown in Table 4.1. Each RGB image in this dataset has its corresponding depth and pose information, that is, RGB-D (RGB-Depth).

Pose Estimation Based on Random Sample Consensus

The scene coordinate regression network mainly regresses the scene coordinates of the corresponding points of the image. After obtaining the estimated scene coordinates, these points need to be used to solve the pose. This problem is very similar to the optimization method, so using the RANSAC algorithm to estimate the pose can obtain robust results. The general implementation steps of the RANSAC algorithm have been introduced in Sect. 4.2.1. In the problem of pose estimation through scene coordinates, some adjustments need to be made, but the core idea is the same. In this technical case, the process of estimating the camera pose using the RANSAC algorithm is as follows:

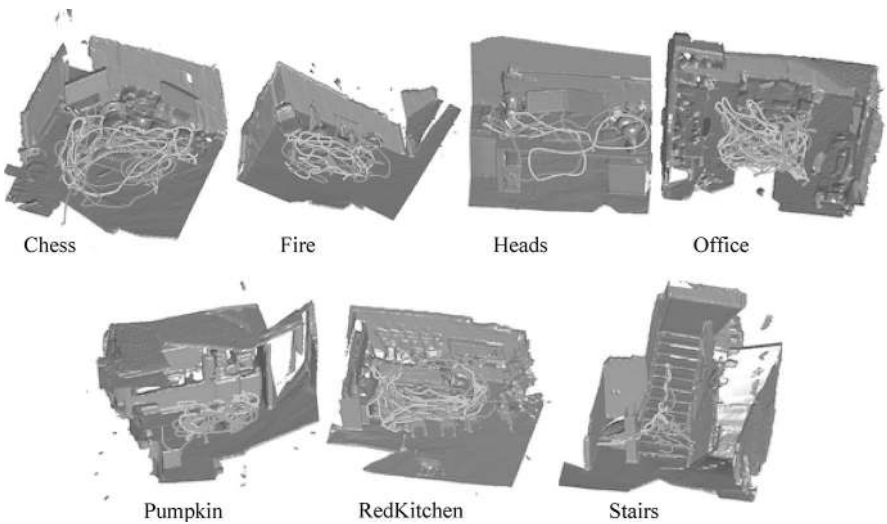


Fig. 4.6 7 Scenes [5] dataset modeling and trajectory diagram

Table 4.1 Distribution of 7 Scenes data [5]

	Within 5 deg.,5 cm	Within 2 deg.,2 cm	Within 1 deg.,1 cm	Median Error	Avg. processing time
Chess	51.30%	12.90%	0.40%	2.0 deg.,4.8 cm	1 s 52 ms
Fire	32.00%	8.10%	0.40%	5.0 deg.,13.5 cm	482 ms
Heads	37.60%	6.30%	0.00%	8.8 deg.,11.4 cm	328 ms
Office	32.60%	4.60%	0.20%	2.6 deg.,8.1 cm	1 s 405 ms
Pumpkin	17.10%	1.80%	0.00%	4.2 deg.,15.6 cm	753 ms
Redkitchen	19.60%	3.00%	0.10%	3.7 deg.,12.6 cm	2 s 82 ms
Stairs	0.40%	0.20%	0.00%	134.8 deg.,383.7 cm	337 ms

- Step one: Randomly select four coordinate points from all scene coordinates to get four sets of 2D-3D corresponding points.
- Step two: Solve the P3P for the obtained four sets of corresponding points. If the solution is successful, record a pose prediction and mark the selected points as selected. Otherwise, repeat the first step.
- Step three: Repeat the first and second steps until n camera pose predictions are obtained.
- Step four: Calculate the reprojection error of the scene coordinates for all pose predictions.
- Step five: Count the number of reprojection errors within the threshold τ , and select the camera pose with the most inliers.
- Step six: Use the inliers of the selected camera pose to calculate a new pose.

Step seven: Recalculate the reprojection error of this camera pose for all predicted scene coordinates.

Step eight: Select the inliers and use these inliers to estimate the pose.

Step nine: Repeat the seventh and eighth steps until the change in the camera's pose is less than σ , and obtain the final camera pose.

The overall estimation accuracy of the scene coordinates determines the prediction accuracy of the camera. The RANSAC algorithm has very strong robustness. Therefore, how to predict more accurate scene coordinates should be a key consideration. Based on this conclusion, this technical case focuses on how to more accurately estimate scene coordinates.

4.3.1.4 Results

The camera relocation method is based on the feature database in 7. The feature database was collected and matched on the Scenes public dataset, and the experimental data results are shown in Table 4.2. From the data, it can be seen that in most scenes, the relocation method based on the feature database can achieve relatively stable results, but in some special scenes, its localization performance is difficult to further apply. In the actual testing process, in order to ensure a certain rate, only part of the image data is used as the feature collection data source for each scene. However, even so, in some scenes with a large amount of data, its localization processing speed is still very slow, and the average processing time of a single frame even exceeds 1 s.

Since the dataset provides RGB-D data and pose information, its data also fully meets the needs of this technical case, and this dataset is also used in DSAC++ [1] and the angle-based loss function [43].

From the experimental data, it can be seen that the camera relocation method based on the feature database can already show good results in small scenes with rich textures and stable and sufficient lighting. However, when the scene is larger, the processing speed significantly decreases, unable to meet the real-time requirements. This is because when the scene is larger, the feature database formed by the

Table 4.2 Experimental results of camera relocation method based on feature database

	Within 5 deg.,5 cm	Within 2 deg.,2 cm	Within 1 deg.,1 cm	Median error	Avg. processing time
Chess	51.30%	12.90%	0.40%	2.0 deg.,4.8 cm	1 s 52 ms
Fire	32.00%	8.10%	0.40%	5.0 deg.,13.5 cm	482 ms
Heads	37.60%	6.30%	0.00%	8.8 deg.,11.4 cm	328 ms
Office	32.60%	4.60%	0.20%	2.6 deg.,8.1 cm	1 s 405 ms
Pumpkin	17.10%	1.80%	0.00%	4.2 deg.,15.6 cm	753 ms
Redkitchen	19.60%	3.00%	0.10%	3.7 deg.,12.6 cm	2 s 82 ms
Stairs	0.40%	0.20%	0.00%	134.8 deg.,383.7 cm	337 ms

scene is too large, and the number of features to be retrieved is too many, making the computation too large, thereby increasing the running time of the algorithm. To improve this performance degradation, the method based on deep learning makes it possible to cope with slightly larger scenes, due to its fixed network model scale, and therefore has a relatively uniform amount of computation.

4.3.1.5 Summary

This section designs and analyzes based on traditional camera relocalization methods and practices the camera relocalization method based on the feature dataset. This section explains the process of this method and introduces the key processes in detail, including the extraction of ORB feature points and the pose-solving method based on RANSAC. These methods are not only applied in this practice but also used in the subsequent process based on deep learning methods, some of the principles of technical methods, etc., so although they are traditional methods, their practical guiding significance is still quite large.

4.3.2 *Case Study: Deep Learning-Based Camera Relocalization*

4.3.2.1 Task Description

In Sect. 4.3.1, the classic camera relocalization algorithm was introduced. Deep learning technology provides new ideas and solutions for scene coordinate regression. The PoseNet model [9] regresses the camera pose through transfer learning, and the GoogleNet model [56] directly regresses the camera's pose, achieving relatively good results. However, for the high localization requirements in the actual environment, this localization method is still very limited, the accuracy is low, and it is difficult to use directly. However, this localization method has started the relocalization technology based on deep learning. Subsequent research has tried to improve its accuracy, but the accuracy of this method is still very low.

In the Differentiable random SAmple Consensus (DSAC) algorithm [3], a VGG deep convolutional network is used to regress the RGB image to obtain scene coordinates, and a differentiable RANSAC algorithm is proposed, so that the entire localization solution process can be trained end-to-end. The way this problem is handled is very similar to the overall idea of random forests, which is mainly to predict scene coordinates.

Scene coordinates are the 3D coordinates of the center 2D point of a small area of the image corresponding to the world coordinate system. The previously mentioned P3P and PnP algorithms can estimate the pose of these corresponding 2D-3D points, but since the calculation of scene coordinates must use depth maps, the entire design process is difficult to avoid using depth information. DSAC++ [1]

proposed an algorithm for scene coordinate regression without the need for a depth map but added an initialization process. In the DSAC++ algorithm, the camera's pose needs to complete three steps of training to get the best pose estimation: the first step is to train using the Euclidean distance between scene coordinates as the objective function; the second step is to further optimize training using reprojection error; the third step is to regress scene coordinates using an end-to-end method. The camera's pose also uses a RANSAC loop that can be added to end-to-end training to obtain. In order not to use depth information, when there is no depth information, DSAC++ The algorithm proposes a heuristic method to generate a fake depth map for training. Since the depth map is obtained heuristically, the first step of training in this case usually does not yield good results. However, the accuracy of pose estimation will significantly improve in the second step using the training step of reprojection error.

The subsequent paper [43] analyzed the reason why training must be done in two steps, because the reprojection error cannot correct points that are behind or too close to the camera center. Therefore, paper [43] proposed an angle-based reprojection error, which can regress scene coordinates without depth and without initializing the network.



Chapter Knowledge Points

DenseNet Network: A fully residual network proposed at CVPR 2017, which ensures that the original information is not lost while ensuring that each backward propagation can feedback to the frontmost network, a network architecture.

Pyramid Pooling Structure: A network architecture that can extract features of different scales and effectively use global information by combining these different scale features, providing help for deep learning to solve global problems.

Reprojection Error: A measurement method to evaluate the accuracy of 3D point estimation by projecting the estimated 3D points onto the image and calculating the Euclidean distance between the projection point and the actual 2D point.

Differentiable Random Sampling Consistent Method: A differentiable random sampling consistent algorithm proposed in DSAC that uses SoftMax or probability methods for candidate pose selection. The introduction of this algorithm allows camera pose estimation methods to be trained end-to-end.

4.3.2.2 Related Work

DenseNet Network

The capabilities of convolutional neural networks are very powerful, but not any convolutional network can achieve good training results. There are some common rules when building convolutional neural networks, such as adding a nonlinear integration layer after the convolutional layer, and generally using a full convolutional layer or a fully connected layer at the end of the network. Blindly designed networks usually do not perform very well, so robust general network structures are a good choice. The network model structures that are now well known include VGG [57], GoogleNet [56], ResNet [58], and DenseNet [59].

VGG [57] was designed by the Visual Geometry Group at Oxford University. It was widely known for ranking second in the ILSVRC competition in 2014, with GoogleNet ranking first. However, generally speaking, VGG performs better in transfer learning tasks. The model structure of VGG is very simple, adding a linear integration function after each layer of convolution, and a maximum pooling layer after every 2–4 convolution layers. In VGG [57], its convolution kernel and pooling kernel are relatively small.

ResNet, also known as residual network, was proposed in 2015 and won first place in the ImageNet competition’s classification task. ResNet50 and ResNet101 are widely used in many methods. The main structure of ResNet is its residual block, as shown in Fig. 4.7. The residual network will pass the previous feature map every few layers for continued use, which can avoid the gradient disappearance caused by the depth of the network being too deep.

DenseNet [59] won the 2017 International Conference on Computer Vision and Pattern Recognition (IEEE/CVF Conference Computer Vision and Pattern Recognized with the best paper honor at the Conference on Computer Vision and

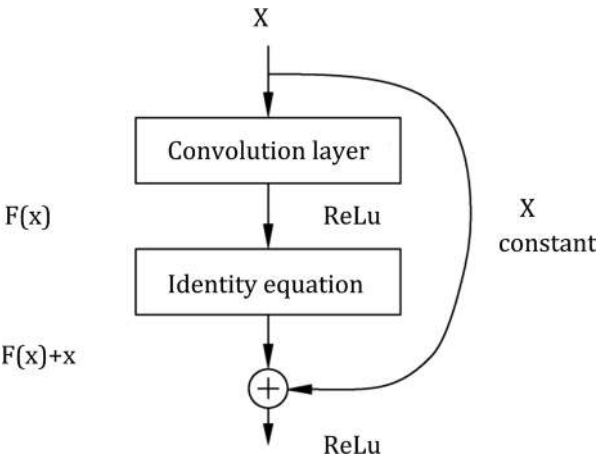


Fig. 4.7 ResNet’s Residual Block

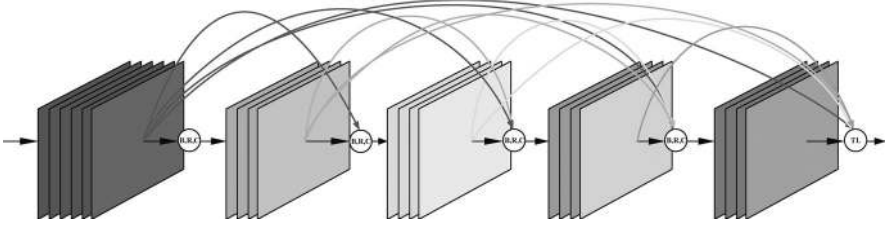


Fig. 4.8 Structure of Dense block

Pattern Recognition (CVPR), one of the major contributions of DenseNet is its dense connection block). It takes the residual concept of ResNet to the extreme, which can be understood as a fully residual network. As shown in Fig. 4.8, it can be seen from the figure that in the structure of DenseNet, the output of each network layer is passed to the subsequent network layers. This operation can be understood as each transmission carries the original information from the front, thus ensuring that the original information is not lost, while ensuring that each backpropagation can directly be feedback to the front network layer. This allows the network to effectively retain most of the information.

It can be said that DenseNet is one of the most stable and effective network model structures in recent years. This network structure used for image processing tasks is also suitable for use in related work on relocalization based on RGB data. If you want to use such a structure in relocalization tasks, you need to analyze DenseNet in detail and make use of it. The most famous models of DenseNet include DenseNet-121, DenseNet-161, DenseNet-169, and DenseNet-201, with the numbers indicating the number of convolutional layers in the model. DenseNet is currently the most superior deep neural network. Considering its excellent performance in image classification tasks, it can be considered that feature extraction through this network model can effectively improve the effect of model training, and in actual performance, it can indeed improve the network's performance. Scene coordinate regression is actually the processing of image tasks. Therefore, in this technical case, DenseNet-201 is used as the feature extraction part of the scene coordinate regression. Due to the limited dataset, this technical case adopts the method of transfer learning, improving the latter part of the transfer learning, that is, adopting the pyramid pooling structure.

Considering that the number of coordinates predicted by the scene coordinate regression network is expected to remain at 4800, this technical case uses a truncated DenseNet as the network module for feature extraction. The truncation position is determined by DenseNet according to the scaling size of the output image. If the output image size is $P_{size} = P_w * P_h$, and the output feature map size is $F_{size} = F_w * F_h$, the scaling factor can be obtained as $S = F_{size}/P_{size}$. Take the reciprocal of the scaling factor S^{-1} . Considering that each Dense block in DenseNet will scale the original image by $1/2$, the number of Dense blocks retained is $\log_2 S^{-1}$. This operation ensures that the result of DenseNet feature extraction can meet the subsequent scene coordinate regression network.

Pyramid Pooling Structure

Pooling is an operation that samples a part of the content specifically, common ones are average pooling and max pooling, these two pooling methods can, respectively, weaken and strengthen features. They are widely used in traditional neural networks and convolutional neural networks. Pyramid pooling can extract features of different scales, and by combining these features of different scales, pyramid pooling can effectively utilize global information, providing help for deep learning to better solve global problems.

The pyramid pooling structure in two-dimensional space is shown in Fig. 4.9. When the feature map is fed into the pyramid pooling structure, it is divided into multiple parallel branches (four branches in this figure), each of which performs maximum pooling on the feature map, resulting in four feature maps of the same width but different scales. These features represent features of different scales, the size of which can be set by yourself. In order to reduce their channel width, these feature maps are then passed through a convolutional network layer to become narrow enough, usually reducing the channel number of the feature map to 1/4 of the original feature map. These feature maps are finally enlarged to the same size as the input feature map using a bilinear interpolation upsampling layer. Since the size of the feature map is consistent with the size before entering the pyramid pooling, these feature maps can be stacked with the input feature maps, thus integrating the global information of the entire scene.

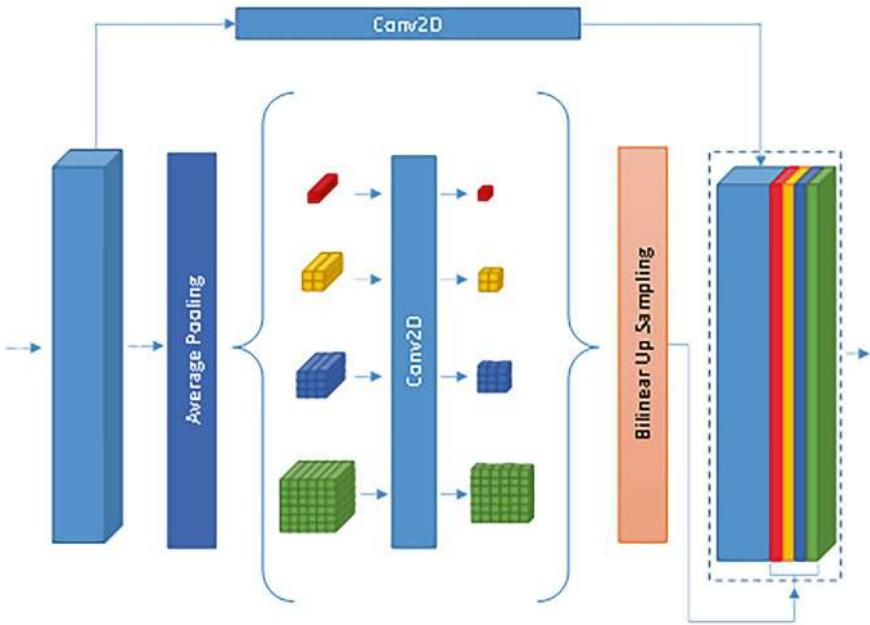


Fig. 4.9 Pyramid pooling structure

This pyramid pooling structure was first published at the top computer vision conference CVPR in 2017. Zhao [60] and others used this structure to design a pyramid scene parsing network,

In pyramid pooling, four sizes are pooled, namely, 1×1 , 2×2 , 3×3 , and 6×6 .

It won first place in the semantic segmentation competition that year, which is also due to the effectiveness of the pyramid pooling network in extracting global information.

Considering the effectiveness of the pyramid pooling network in integrating global information, this module is used as part of the deep learning network structure in the network design of this technical case. Experiments can prove that sufficient and effective global information can improve the effectiveness and performance of the network. The experimental and verification part will be introduced in the results display later.

Objective Function

This technical case uses the angle-based reprojection error as the objective function, but before describing the angle-based reprojection error, let's first introduce the reprojection error. The reprojection error can be understood as the distance between the actual image coordinates and the coordinates of the 3D point projected back to the image coordinate system under the world coordinate system, as shown in Fig. 4.10. Where point p is the world coordinate system, its actual imaging position in the two image planes is p_1 and p_2 , but according to the estimated camera pose on the right, the projection of point p on the right is at point p'_2 , so $\|p_2 - p'_2\|$ is the reprojection error, its specific form is

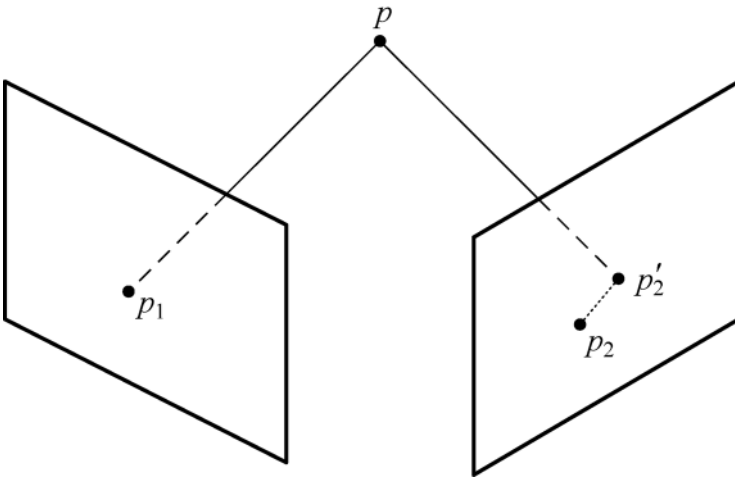


Fig. 4.10 Schematic diagram of reprojection error

$$Loss_{\text{reprojection}} = \sum_k \|CP^{-1}p_k - p'_k\| \quad (4.1)$$

where C is the camera internal parameter, P is the estimated camera pose, k represents the number of 3D points, p_k represents the scene coordinates, that is, the point p shown in Fig. 4.10, and p'_k represents the actual 2D coordinate value. The entire reprojection error describes the relationship between the scene coordinates and the camera pose. They need to cooperate with each other to make the error as small as possible. Since the reprojection error only describes the difference between 3D points and 2D point pairs, a pose has to take care of all the corresponding relationships at the same time, which is very difficult to achieve at present, so it can only make the reprojection error of a picture as small as possible, rather than let the pose make the reprojection error of the whole picture 0.

Brachmann et al. [1] used reprojection error as the objective function for the convolutional network, but to use this objective function, some depth assumptions must be made to initialize the network. Li etc. [43] pointed out that the design of the reprojection error is to reproject all images back to the image coordinate system, but some special cases were not considered during the design. In these cases, the reprojection error will fail. As shown in Fig. 4.11a, where x' represents the estimated scene coordinates, c represents the camera center, x represents the actual scene coordinate points, and x' and x represent the projections of the estimated scene coordinate points and the actual coordinates, respectively. It can be seen that if the scene coordinates are behind the camera center, the reprojection error can still be

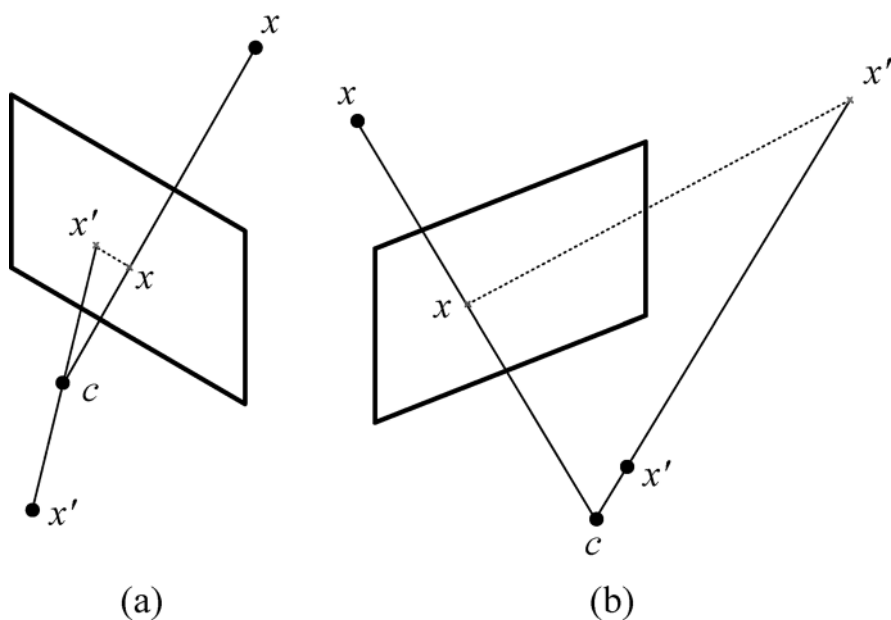


Fig. 4.11 Failure of reprojection error

very small, even if these two points are completely different points with a huge distance difference. In this case, the reprojection error will be useless. Figure 4.11b shows another situation where the reprojection error fails. When the predicted point of the scene coordinates is very close to the camera center, the depth value of the scene coordinates may be a very small number, causing the reprojected 2D coordinates to deviate greatly from their actual position. These two failure situations ultimately lead to the network's inability to converge stably [1], so network initialization is necessary. To avoid unnecessary calculations, it is proposed in [43] to use angle-based reprojection error as the objective function.

The angle-based reprojection error is shown in Fig. 4.12. Because the simple reprojection error as a loss function will lead to non-convergence or stop at a local minimum due to the aforementioned defects. Therefore, if you want to use reprojection error, you need to initialize the network and control all predicted scene coordinates in front of the imaging plane. Figure 4.12 describes the angle-based reprojection error, considering projecting the point x' in the image coordinate system back to the camera coordinate system with a depth of focal length f to get the point x'_θ . Next, draw a sphere with the distance from x'_θ to the camera center c as the radius, and the estimated scene coordinate x is projected onto this sphere proportionally. If they do not coincide, an angle θ will be formed. To minimize the angle θ , you can minimize $\|x_2 - x'_\theta\|$. The complete formula is

$$Loss_{ang} = \sum k \in P_i \left\| \frac{\|d_k\|}{\|D_k\|} P^{-1} p_k - fC^{-1} p'_k \right\| \quad (4.2)$$

where $d_k = fC^{-1}p'_k$, $D_k = P^{-1}p_k$, f is the focal length, C is the camera intrinsic matrix, P is the estimated camera pose, p_k represents the scene coordinates, i.e., the point x in the figure, and p'_k represents the actual 2D coordinates, i.e., the point x' in the figure.

As can be seen from Fig. 4.12, the angle-based reprojection error actually projects both the scene coordinate points and the image coordinate points onto a sphere with a depth of focus, and then measures their spatial Euclidean distance on the sphere. If these two points are very close, then their effect is the same as the reprojection error. This design can avoid the two problems that the reprojection error will encounter.

Model Framework

This technical case uses a pre-trained DenseNet as the invariant part in transfer training, and the part that participates in training uses a pyramid pooling module, as shown in Fig. 4.13. The construction of this network model is relatively easy. However, to train a deep convolutional neural network, an objective function needs to be set. The objective function is very important for the neural network. The purpose of the entire neural network is to adjust the parameters of the entire network so that the value obtained by the objective function is minimized after the input data is

Fig. 4.12 Angle-based reprojection error

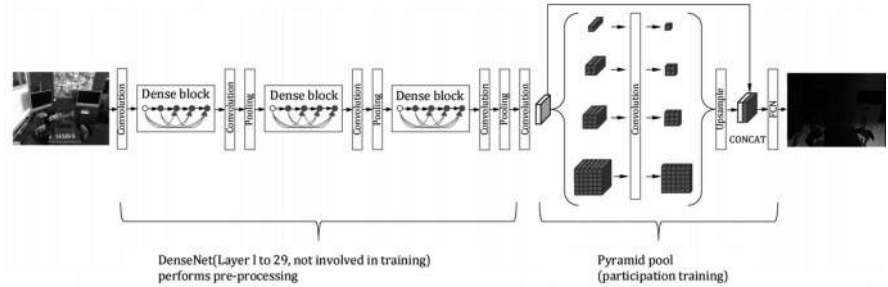
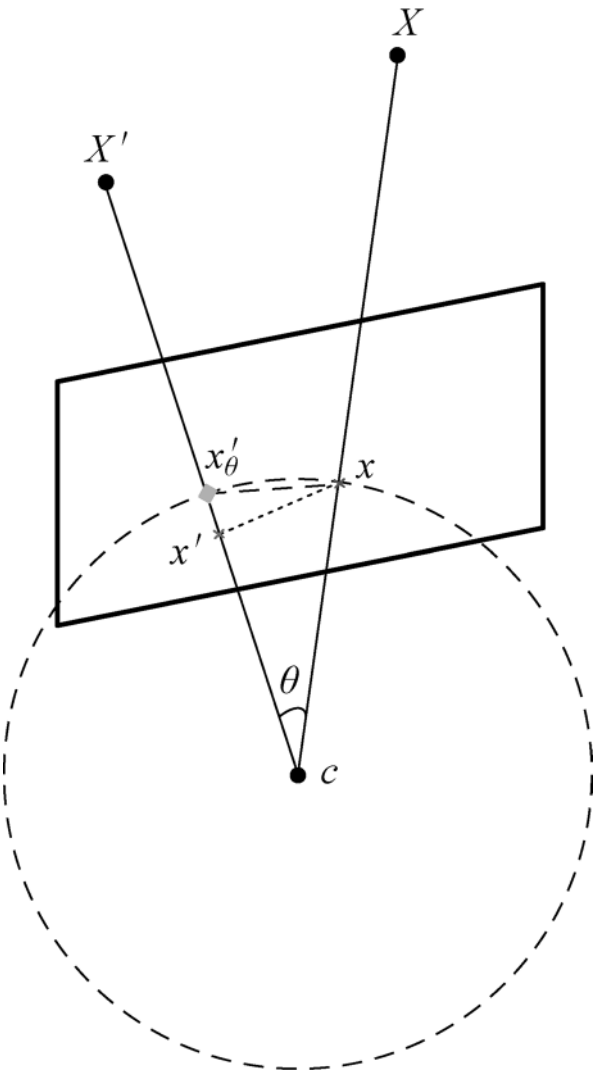


Fig. 4.13 Scene coordinate regression model

calculated. It can be seen that the objective function plays a very important role in the neural network.

A reasonable adjustment to the objective function can further enhance the performance of the network. This technical case adds some innovative weights to the previously proposed objective function to better train the neural network. Two main weights are designed, and their effectiveness can be proven through experiments, which will be described after the description of the basic loss function.

4.3.2.3 Technical Method

The relocalization process based on deep learning methods: The camera relocalization method based on deep learning methods is divided into two steps. The first step is to regress the scene coordinates. In Sect. 4.2, the representation of the camera pose and the solution of the camera pose are introduced. The 3D coordinates in the 2D–3D correspondence used in the P3P and PnP pose solution methods mentioned can be understood as the scene coordinates described here. The difference is that the 2D in the deep learning method refers to the center of a picture block, and the estimated 3D coordinates also correspond to the coordinates of this center in the world coordinate system.

After obtaining the scene coordinates, the second step is to solve the pose. Since the coordinates are estimated, the overall scene coordinates are not precise and need to be solved by optimization algorithms. Considering that there are a large number of outliers in the coordinate pose estimation, which will greatly affect the accuracy of the scene coordinate estimation, the random sample consensus (RANSAC) algorithm is usually used for iterative calculation of pose estimation and optimization.

Relocalization Process

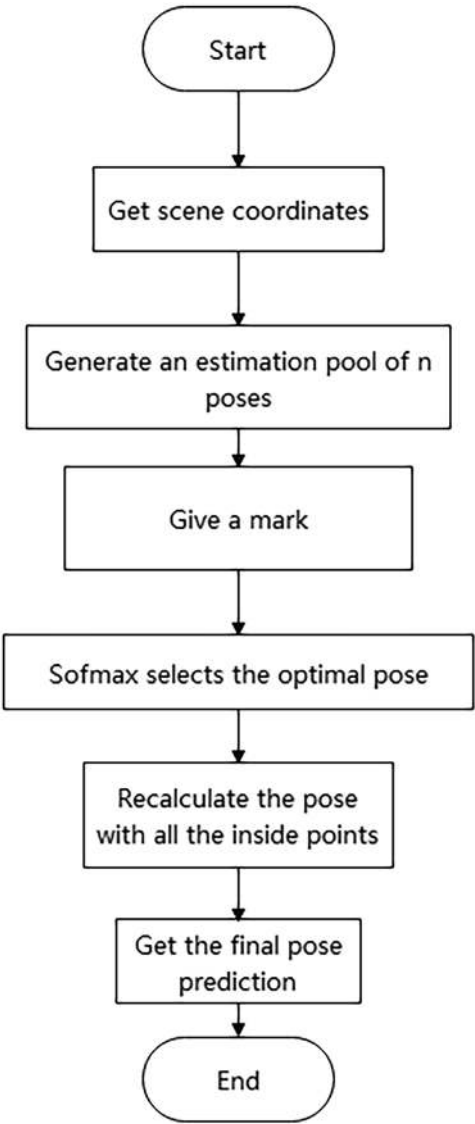
Figure 4.14 shows in detail the localization process adopted in this book, obtaining coordinate point predictions through the network model designed in this book. The second step is broken down into several specific steps. First, as mentioned in the previous RANSAC algorithm, a pose estimation pool containing n pose estimates is generated, and then these pose estimates are scored. The score first calculates the reprojection error of each pose for all points and then sums them after processing by the sigmoid function. The SoftMax function is used to select the best one among these poses. Before SoftMax, the score difference is small, but after the SoftMax function operates on the data, the highest score is easier to select. After selecting the pose with the highest score, select their inliers for the RANS.

AC algorithm iterative process, finally obtaining the predicted pose.

4.3.2.4 Results Display

After completing all the preparatory steps, you can start training the scene. After programming the dataset importer in C++, use Torch7 to build the model and train it. During the training process, randomly select images from the training set to train

Fig. 4.14 Based on deep learning method



the convolutional network, use the Adam optimization algorithm, the initial learning rate is 10^{-4} , and then the learning rate is halved every 25,000 iterations after 50,000 iterations. The model is saved every 1000 iterations, and a total of 200,000 iterations are trained. The loss function used during training is the angle-based reprojection error with weight. To test the impact of weights on experimental results, each weight is trained separately. The time it takes to train and test a model on one GPU for a scene is 24–48 h.

4.3.2.5 Summary

“Knowing” one’s location is a key technology for many applications, such as autonomous driving and robots. Traditional localization technology, due to its frame-based matching, has a high error rate in pose calculation, and the error will accumulate over time, leading to a significant deviation between the later pose trajectory and the actual trajectory. In addition, due to the limitations of feature extraction, the calculation of pose requires a high quality of images, and relocalization technology was born to avoid these problems. This technical case can achieve a higher accuracy of camera relocalization based on RGB input, which can better accurately and efficiently locate images in environments with prior scene knowledge. The full-text starts from localization, discusses the representation and basic methods of localization in 3D scenes, introduces relocalization, describes the latest relocalization methods, and proposes to use DenseNet-201 as a pre-training network, using pyramid pooling to integrate global information, using a weighted angle-based objective function to estimate scene coordinates without depth, and obtaining camera poses. The experimental results show that these improvements have a certain improvement in the accuracy of relocalization. Although current deep learning methods can perform scene coordinate regression and obtain better camera pose estimation with less information, the performance of deep learning methods in some challenging scenes is still very limited, such as scenes with a large amount of repeated textures and scenes with a lot of white. In addition, the accuracy of deep learning methods is still not as good as the method of random forest regression scene coordinates, but the method of random forest must be supported by depth information. There are still some limitations to the methods based on deep learning, considering that redundant content can be refined and analyzed in the future to distinguish their differences. In addition, the current ability to regress scene coordinates is still very limited, and the effect is not very satisfactory, whether it is accuracy or generalization problems are very limited, to achieve a higher accuracy of scene coordinate regression is the key to achieve more accurate camera relocalization. Therefore, how to have a better expression of images may be a direction to consider, and the generalization ability of the network is also a direction to consider. In addition, all current machine learning methods require training, and different methods have different training times. For the deep learning method in this technical case, the training time for each scene is measured in hours, which is enough to understand that the time consumption of this method is very large, so there is still a long way to go for practical applications, so considering how to efficiently obtain the network training model is also a direction for future research.

4.3.3 Case Study: Robot Navigation Based on Camera Relocalization

4.3.3.1 Task Description

Autonomous localization navigation is one of the cores of the robot intelligent system, and it is a key influencing factor to endow robots with the ability to perceive the environment and move. In order to make the robot have good localization and navigation effects in indoor scenes and have a certain universality, this section starts from the perspective of computer vision, carries out robot localization based on neural networks, combines with the robot system to complete the robot localization navigation task, and utilizes it in actual scenes, thereby verifying the practical value of the system method. Through visual sensors, robots can obtain richer scene environment information, solving the problem that laser sensors can only obtain less two-dimensional environment information, and at the same time reducing the cost of use. The localization part adopts the localization method based on the full convolutional neural network to densely regress the scene coordinates, defines the correspondence between the image and the scene space, that is, scene localization, and realizes the function of the robot to locate itself by obtaining a single RGB image under the current view. The result of localization plays a key role in the robot's movement and planning the path to the destination coordinates. Combined with the robot mobile platform system, the robot navigation task is finally completed.



Chapter Knowledge Points

Building Maps: The process of identifying spatial obstacles can be divided into three types according to performance and expression methods: grid maps, node maps, and hybrid maps.

Path Planning: The algorithm for robot operation path planning based on the map and the sensor's perception of the environment is divided into static path planning and dynamic path planning.

Dijkstra Algorithm: Also known as the shortest path algorithm, it is a method for finding the shortest path between two nodes. It adopts the idea of breadth-first search and expands layer by layer to the endpoint.

4.3.3.2 Related Work

ROS

ROS is an open-source operating system for robot function development, including a set of libraries, packages, and development tools. The most valuable thing is to achieve code reuse on different robot platforms without changing the code.

ROS is mainly built on the Linux system and can provide many functions similar to traditional operating systems, such as hardware abstraction, low-level device control, common function implementation, inter-process message passing and forwarding, and program software package management.

ROS includes a three-layer architecture, namely, the file system level, the computation graph level, and the open-source community level. The internal structure of ROS, the file structure, and the required core files all belong to the file system level. The open-source community level is mainly used to obtain and share ROS resources and is an independent online communication community. The implementation of complex robot functions cannot be separated from communication between processes or nodes. During the execution of ROS, all related participating devices, sensors, controllers, and actuators are abstracted into single nodes. The communication medium between processes or nodes is a communication network created by ROS. This network connects all process nodes, and any node in the system can communicate and interact with the nodes to be contacted through this network, and can publish its message data to the communication network for other process nodes to obtain, which belongs to the computation graph level, that is, a point-to-point network communication method [23, 24].

Building Maps

Building a map is actually a process of finding spatial obstacles. At present, the forms of maps built according to different performances and expression methods are mainly divided into three types, namely, grid maps, node maps, and hybrid maps [61, 62]. The following is a brief introduction to the three types of maps:

- (1) **Grid Map:** As the name of the map describes, this type of map represents the environment around the robot with many small squares of the same size during the construction process. If there are obstacles, they are marked as 1, otherwise, they are 0. But in actual applications, in order to improve the accuracy of map representation, the judgment of obstacles is no longer directly converting the signal returned by the sensor once into 0 or 1, but converting it into a probability value to represent the possibility of obstacles. Combine the probability values returned by the sensor several times to finally determine whether there are real obstacles. This method effectively avoids sensor errors, the constructed map is relatively accurate, and the representation of the surrounding environment is also relatively complete [25]. It is more suitable for the construction of medium or small scene environment maps. The map construction part in this technical case uses a grid map.
- (2) **Node Map:** Node maps are more suitable for large or even super-large scene environments. They represent local environments with the same characteristics with one point. The entire map is a node map, such as each city in the map of China is represented by one point. Compared with grid maps, the biggest advantage of node maps is to save memory resources and computing resources, but they are far from enough in terms of detailed description.

- (3) **Hybrid Map:** The hybrid map combines the advantages of grid maps and node maps. The local environment adopts the form of a grid map, and the global environment adopts the form of a node map. From a theoretical point of view, it is a combination of complementary advantages, but the biggest problem at present is how to effectively integrate the two types of maps.

The path planning component is a key part of autonomous navigation technology, determining whether a robot can accurately start from the initial state, avoid obstacles, search for an optimal or sub-optimal collision-free path, and reach the target state. Path planning can be divided into static path planning and dynamic path planning based on whether the environmental information is fully known [28, 29].

Path Planning

- (1) **Static Path Planning:** This method is based on the premise that the scene environment information is fully known, also known as offline global path planning, such as map construction can accurately represent the structural information of the scene. The global planner determines the optimal route for the robot based on the robot's localization results, target point information location, and scene map structure, aiming to find the optimal movable path.
- (2) **Dynamic Path Planning:** This method is based on the premise that the scene environment information is partially or completely unknown, also known as online local path planning. During the robot's movement, the robot's perception system is also continuously working. When a dynamic object or unknown obstacle is detected, if the obstacle avoidance behavior has a higher priority than following the original route, the local planner will replan the path based on the updated environmental information, driving the robot to move according to the local planning results, effectively avoiding possible obstacles and reaching the target state. The path planning part of this technical case uses dynamic path planning, making it more friendly to slight changes in the scene environment.

4.3.3.3 Technical Method

The previous work completed the first step of the autonomous navigation system, determining the indoor scene environment of the robot's operation and the robot's indoor scene location, answering the question "where am I." The following answers the question "how do I get there," that is, the process of mobile navigation. The path planning of navigation is introduced and practiced.

In ROS, the robot's global planning defaults to using the NavFn package for navigation, which uses the shortest path algorithm (Dijkstra algorithm) for path planning. Dijkstra mainly finds the shortest path between two nodes, using the idea of breadth-first search, expanding from the starting point layer by layer until it reaches the end point. This technical case introduces the steps of path planning using the idea of a weighted graph data structure.

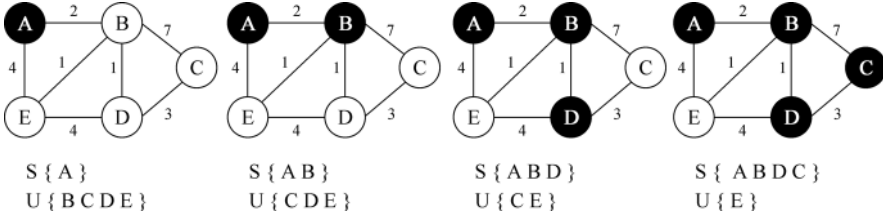


Fig. 4.15 Dijkstra algorithm to find the shortest path

The steps of path planning are as follows, Fig. 4.15 shows the process of Dijkstra's search for the shortest path.

- (1) Specify the starting node A, that is, the current location and rotation of the robot.
- (2) Set two sets S and U, where S stores nodes that have found the shortest path, and U contains nodes that have not found the shortest path.
- (3) Initialize S and U, where S only contains the starting node A, and U contains other nodes in the weighted graph.
- (4) Compare the path distance of nodes in set U with set S, and add the node with the shortest path to set S.
- (5) Update the set, repeat steps (4) and (5) until the shortest path from the starting point to the target point is obtained. As shown in the figure below, A is the starting point, D is the target point, showing the process of finding the shortest path using the Dijkstra algorithm.

4.3.3.4 Results Display

Figure 4.16 shows a basic example of robot localization and navigation. According to the system flowchart, the current RGB image under the current viewpoint is obtained through the robot's visual sensor, and the current robot pose is generated using the localization model. The generated pose is then transformed according to the actual settings. In this technical case, the pose generated by the localization network model is represented as a rotation matrix and a translation vector. When setting the pose, the quaternion form is used to replace the rotation matrix, which has better readability. Therefore, it includes the process of converting the rotation matrix to quaternion. Then, based on the localization results, the pose of the robot Tiago on the map is corrected, the target position is given, the path planning algorithm is called, and the robot is driven to perform autonomous navigation tasks, reach the specified position, and then perform subsequent interaction tasks. In this technical case, the robot's grabbing is used as an example to verify the system's obstacle avoidance performance. The planned path and the actual movement trajectory of the robot can prove that this technical case localization navigation system can reach the set goal and has certain practical value.

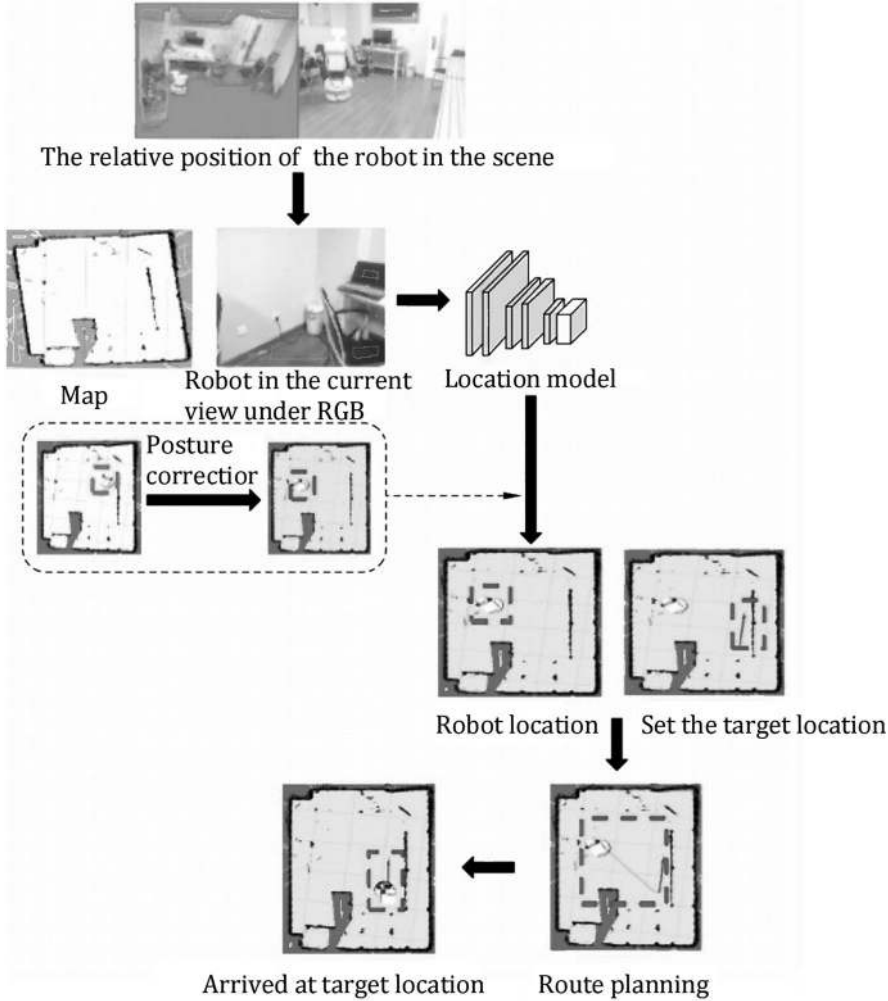


Fig. 4.16 Localization and navigation experiment (no obstacles)

4.3.3.5 Summary

This section uses the Tiago intelligent mobile robot platform to complete robot interaction tasks. In this section, the system goal is to complete robot indoor localization and navigation interaction tasks based on vision. Robot localization is the core of the entire system. The current localization method based on image frame matching is not friendly to indoor scene environments, prone to pose loss, and error accumulation can lead to drift and localization failure. Moreover, the high dependence on depth information and high computational complexity makes it impossible

to meet the real-time requirements of localization, and the success rate in actual applications is low.

Based on the above problems, this technical case combines the advantages of different methods and designs a stable and efficient robot localization and navigation system. From the perspective of vision, neural networks are used for localization, and the idea of image frame matching is used to obtain a small amount of pose and image information for the initialization and training of the localization neural network model. A small amount of data information can effectively avoid the occurrence of pose loss and drift. Using limited dataset information to train and generate an indoor scene localization model, based on the idea of scene coordinate regression, it is independent of depth information. Using the function of single RGB image localization, the generation of the indoor localization model can be performed independently offline, so during the system use process, this localization model can be directly called, and localization information can be generated in real time without a large amount of computation, meeting the task's requirements for system accuracy and real time. The localization results are combined with the indoor map constructed by laser radar scanning and the information of the moving target position, and the Dijkstra algorithm is used for path planning to control the robot Tiago to complete the navigation task as planned.

The system aims to fully utilize visual information to achieve robot indoor localization and navigation interaction tasks, combining the advantages of current mainstream methods, using multiple sensors to achieve a high-precision, high-efficiency, and strong stability localization navigation system. Among them, the localization method based on neural networks makes up for the shortcomings of the simple image frame matching localization method that is prone to localization failure and cumulative error or even drift. At the same time, the method based on image frame matching creates conditions for the initialization of the localization neural network model; Visual sensors obtain a large amount of environment and structure information, making up for the deficiency that laser radar can only obtain information on its own plane, and at the same time, laser radar improves the accuracy of visual localization navigation. The combination of the two can effectively expand the detection range and improve information accuracy.

4.3.4 Case Study: Robot Obstacle Avoidance Based on Known Grid Map

4.3.4.1 Task Description

The path planning algorithm of the ROS robot navigation package is based on a known map. It requires the use of SLAM technology to first construct the map file and publish it to the robot, enabling it to calculate based on this map, thereby realizing specific navigation and obstacle avoidance. ROS has already provided a navigation package based on a known map, using the A* algorithm to implement global

path planning and the DWA algorithm to implement local path planning. This technical case introduces the principles, configuration methods of the gmapping package used in SLAM mapping, and the implementation details, principles, and parameter adjustments of the navigation package.



Chapter Knowledge Points

Laser radar (Lidar): It is a radar system that detects the position, speed, and other characteristics of the target by emitting laser beams. Its working principle is to emit a detection signal (laser beam) to the target, and then compare the received signal reflected back from the target (target echo) with the emitted signal. After appropriate processing, relevant information about the target can be obtained.

Rviz: It is a 3D visualization platform in ROS, which can not only realize the graphical display of external information but also publish control information to objects through rviz, thereby realizing the monitoring and control of robots.

Manhattan distance: The distance between two points in the north-south direction plus the distance in the east-west direction, that is,
 $d(i, j) = |x_i - x_j| + |y_i - y_j|$.

4.3.4.2 Related Work

Navigation Map Construction

- (1) SLAM technology: Autonomous navigation is often inseparable from SLAM, which can be described as: The robot estimates its position and constructs a map by moving in an unknown scene, achieving autonomous localization and real-time mapping.

SLAM can use RGB-D cameras to map, or it can use Lidar to map. Due to the slow processing speed of image information and 3D point cloud information, this technical case uses the two-dimensional data of Lidar to implement SLAM mapping. It is very convenient to implement robot SLAM mapping in ROS, and there are many packages for developers to use, such as gmapping, hector_slam, cartographer, and rgbdslam. This technical case uses the gmapping package provided in ROS to create maps.

The gmapping package uses a particle filter algorithm, and the overall framework of gmapping is shown in Fig. 4.17.

- (2) Map construction based on Lidar: Create a virtual world in the Gazebo simulation environment as the environment for map construction. Then create the launch file of gmapping, import the gmapping package, check whether the parameters in the configuration file of gmapping match, the more important

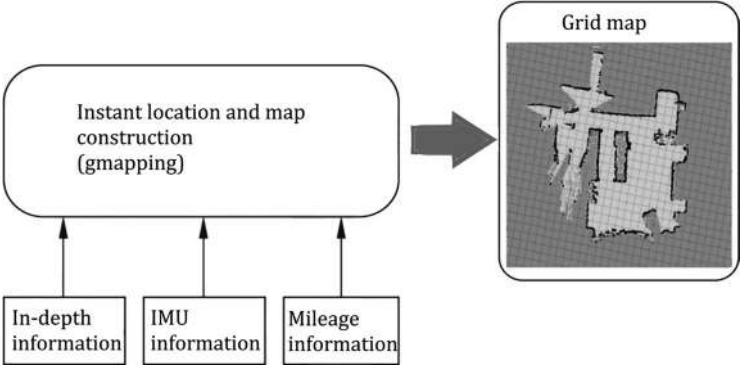


Fig. 4.17 Overall framework of the gmapping package

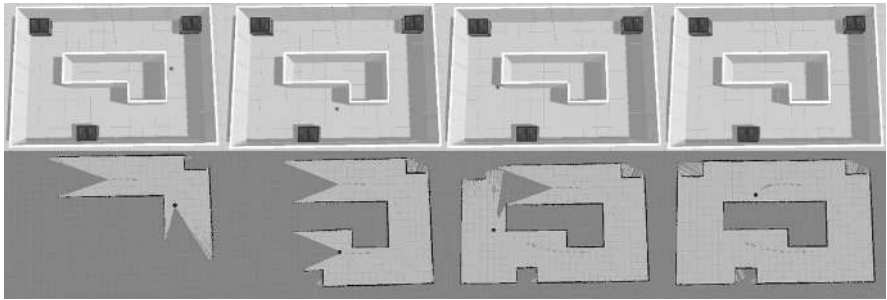


Fig. 4.18 Simulation process of gmapping SLAM based on Lidar

ones are the `odom_frame` and the subscribed laser topic, correspondingly check the odometer and laser topic published by the robot, whether the names are consistent, if not, use the tag for remapping.

Start the RVIZ visualization simulation software, you can observe the process of map construction. Then start the script file that controls the robot to move with the keyboard, and manually control the robot to traverse the map to build a grid map.

The mapping process in the simulation environment is shown in Fig. 4.18. The upper half is the virtual environment created in Gazebo. Use the keyboard control script to control the robot to move in this environment, and use the gmapping package based on Lidar to construct the map. The lower half shows the process of the map gradually forming in RVIZ. After the map construction is completed, use the `$roslaunch map_server map_saver` command to save the constructed grid map and the map configuration file to the current folder, in preparation for subsequent navigation.

Navigation Framework

ROS provides a complete navigation framework based on the `amcl` localization package and the `move_base` navigation package. The data from the robot's onboard sensors must be provided to this package to achieve obstacle avoidance effects, such as laser radar and RGB-D cameras. It also needs to subscribe to odometer information to achieve localization. By receiving the coordinates of the target point, it plans the globally optimal path in the known map, and then publishes the local speed control topic based on this path and sensor information, thereby enabling the robot to achieve obstacle avoidance and navigation.

4.3.4.3 Technical Methods

Global Path Planning Algorithm (A* Algorithm)

- (1) Principle and process of the A* algorithm: The navigation package in ROS provides two algorithms, Dijkstra and A*, to implement global path planning. Since the efficiency of the A* algorithm is significantly higher than that of the Dijkstra algorithm, this technical case uses the A* algorithm to implement basic navigation.
- (2) Specific implementation of the A* algorithm in ROS: The navigation package in ROS has opened a service called `makePlanService`. Through the plugin mechanism, it provides an interface to subscribe to the required topics and services, accepts map information published by the `map_server` node, target point coordinates and angles published by the `move_base_simple/goal` node, joint information and TF coordinate conversion published by the robot model, and odometer information required for `amcl` localization. It can generate path information and visualize this path in RVIZ.

The open table in ROS is stored in the structure of a heap (complete binary tree). Each time a value is taken from the open table, a small top heap sort is performed to take out the top element of the heap. There is no close table in the algorithm. At the beginning, the f value of all nodes except the origin is set to infinity. As long as the f value of this point is not equal to infinity, it indicates that this point has been searched, thereby replacing the close table.

Using Manhattan distance, the distance of a small grid in the grid map is set to 50 by default. The published map configuration file describes the necessary parameters of the map and the complete map information saved in the form of pictures. There are only two colors in the map, black and white, which are converted into a matrix containing only 0 and 1 values to publish map information. The A* algorithm is based on this matrix to perform global path planning.

Local Path Planning Algorithm DWA

The local path planning of the navigation package in ROS adopts the dynamic window approach (DWA). This method samples multiple groups of speeds in the speed space (v, w), simulates the use of this speed for a period of time, checks whether a collision will occur, and then publishes the optimal speed to the robot. The core idea is to convert the local path planning problem into an optimization problem in the speed space (v, w).

- (1) Robot motion model: Movement is relative to the robot. It is assumed that the robot moves in a straight line in a very short period of time (actually it should be an arc, approximated as a straight line). The displacement $\Delta x \Delta y$ can be obtained by mapping to the world coordinate system, so the robot's motion trajectory can be obtained.
- (2) Speed sampling: Due to the limitations of the robot's own hardware equipment, the speed sampling must be within a certain range. The algorithm samples within this speed range, simulates the motion trajectory in a period of time, detects the position of the nearest obstacle, calculates the distance, and then calculates whether it will hit the obstacle along the current trajectory, or whether it can stop before hitting the obstacle. If it can, this speed is acceptable, otherwise it is discarded. Speed sampling is shown in Fig. 4.19.
- (3) Evaluation function: Evaluate the obtained multiple trajectories. Evaluation function:

$$G(v, w) = \sigma^{[\alpha \cdot \text{heading}(v, w) + \beta \cdot \text{dist}(v, w) + \gamma \cdot \text{velocity}(v, w)]} \quad (4.3)$$

- (1) Angle evaluation function: **heading**(v, w) is used to evaluate the difference between the angle of the robot after traveling at the current speed and the final angle of the published destination. It is evaluated by the method of $180^\circ - \theta$, the smaller the θ , the higher the score. As shown in Fig. 4.20.
- (2) Gap: **dist**(v, w) is the distance between the robot and the nearest obstacle on the current trajectory.
- (3) Speed: **velocity**(v, w) is used to evaluate the magnitude of the speed of the current trajectory.
- (4) Smoothing: Normalize the **heading**, **dist**, and **velocity** by dividing each item by the total of all samples of that item, and then adding them up.

$$\text{normal\_head}(i) = \frac{\text{head}(i)}{\sum_{i=1}^n \text{head}(i)} \quad (4.4)$$

$$\text{normal\_dist}(i) = \frac{\text{dist}(i)}{\sum_{i=1}^n \text{dist}(i)} \quad (4.5)$$

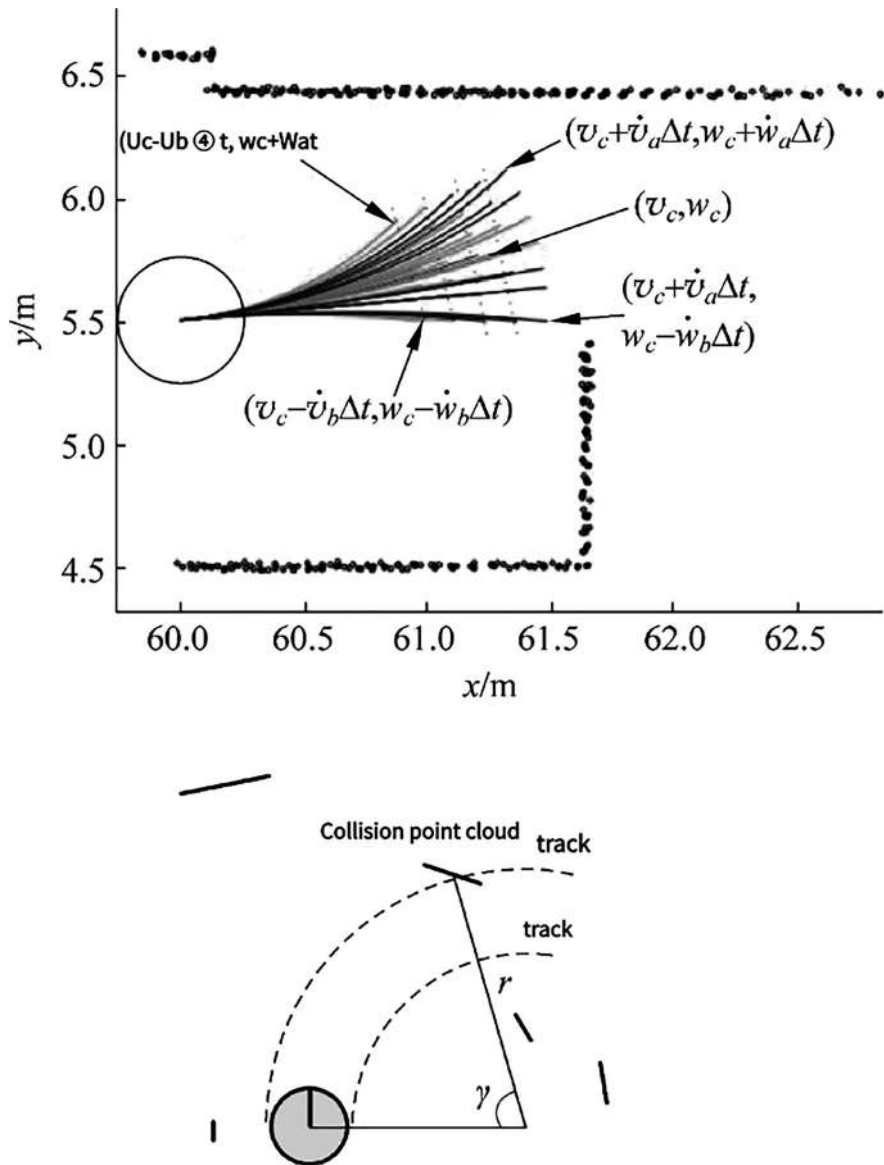
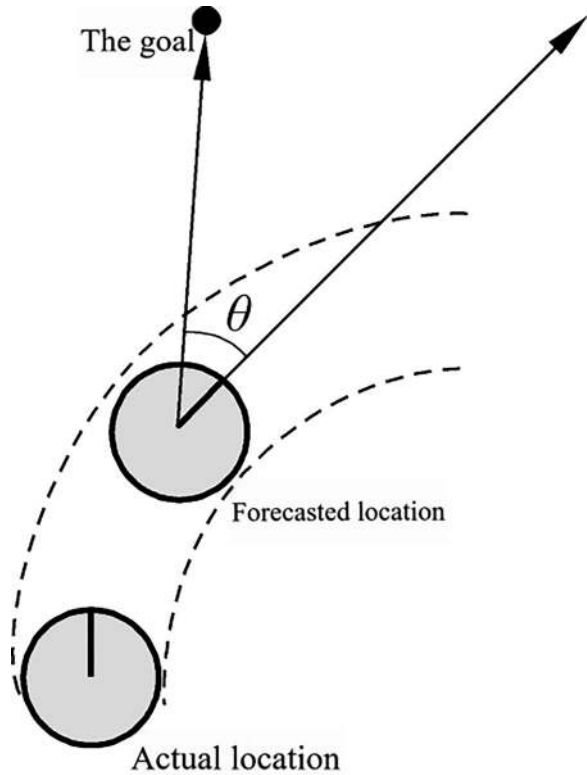


Fig. 4.19 DWA algorithm speed sampling

$$normal\_velocity(i) = \frac{velocity(i)}{\sum_{i=1}^n velocity(i)} \quad (4.6)$$

where n is all the sampled trajectories, and i is the current trajectory to be evaluated. The purpose of normalization is to smooth and prevent one item from dominating in

Fig. 4.20 Azimuth difference



the evaluation function and ignoring other items. By normalizing, they all become the same percentage. The meaning of the evaluation function: The robot arrives at the destination at a faster speed while avoiding obstacles.

4.3.4.4 Results Display

Launch the controller, sensor, and other hardware devices of the car in the host, first implement the function of the ROS navigation package, build a map in the current indoor environment,

Using the gmapping package, after the map is built, import the map into the move_base package, configure the parameters, and publish the target point coordinates, the car can reach the target position, but in the case of obstacles, the obstacle avoidance is time-consuming and inefficient, and the time spent hovering in front of the obstacle is too long, as shown in Fig. 4.21.

Fig. 4.21 ROS navigation package real navigation demonstration



4.3.4.5 Summary

In summary, by obtaining laser radar information, determining the speed range that the robot can actually reach according to the current speed and model, sampling within this range, calculating the distance to the nearest obstacle with the sampled speed and laser radar information, and comparing it with the braking distance, if it can stop in time, calculate the parameters in the evaluation function and calculate the score, and finally take the highest score of linear speed and angular speed LIZI robot path planning debugging and implementation.

4.3.5 Case Study: Multi-Agent Obstacle Avoidance Based on Deep Reinforcement Learning

4.3.5.1 Task Description

The main design tasks are as follows:

- (1) Establish an abstract model of the agent's path navigation, which must have high generality and portability.
- (2) For the model established in step (1), design and implement a robot path planning algorithm that can find the path strategy π_{θ}^* to control each agent.
- (3) The path navigation system is required to complete the simulation path navigation of the agent in the virtual environment and optimize the path of π_{θ}^* .

The main design tasks of this technical case are shown in Fig. 4.22.

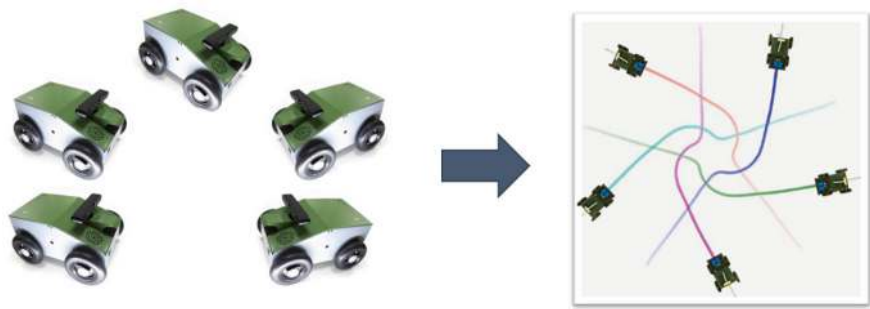


Fig. 4.22 Main design task diagram



Chapter Knowledge Points

Gazebo Simulation Environment: Gazebo is a powerful 3D physical simulation platform with a powerful physical engine, high-quality graphics rendering, convenient programming and graphic interface, and most importantly, it is open source and free.

Unified Robot Description Format (URDF): URDF is a format based on XML specifications for describing robot structures. According to the designers of this format, the purpose of designing this format is to provide a specification for robot descriptions that is as universal as possible.

4.3.5.2 Related Work

Introduction to Lizi™ Robot

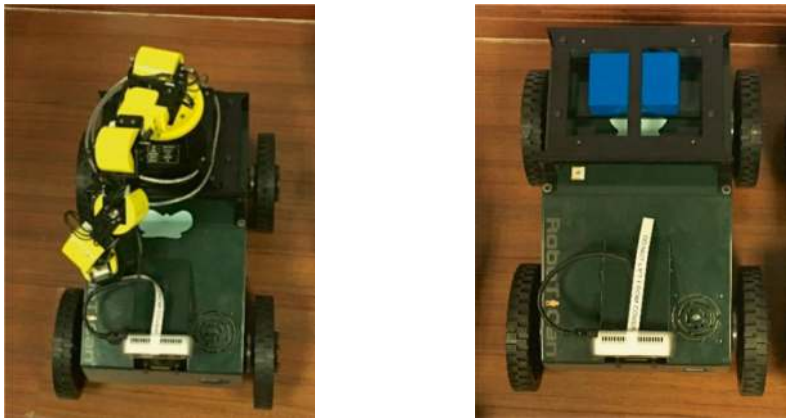
Lizi™ is a robot platform designed and manufactured by the Israeli company Robotican [57], specifically designed for good indoor and outdoor operation. The Lizi™ robot uses skid steering to achieve maximum maneuverability and is small and lightweight, suitable for indoor and outdoor movement and navigation. The Lizi™ robot is sturdy and durable, small in size, and compact in structure. The robot is equipped with a full set of sensors for autonomous navigation and can use mature ROS packages. The Lizi™ robot, specially designed and programmed, can be used for individual work or as part of a group of Lizi™ robots. Table 4.3 is the detailed specifications of the robot.

Hardware Design Parameters

This technical case involves experiments completed on an experimental platform composed of a high-performance host and several Lizi™ robots. The Lizi™ robot has been customized on the official default model, with an additional mechanical arm installed, which increases its own weight, as shown in Fig. 4.23.

Table 4.3 Lizi™ robot hardware parameters

Parameter	Numerical value
Long(cm)	20
Width(cm)	31
High(cm)	38
Weight(kg)	9.1
Maximum velocity(km/h)	3.6
Maximum load(kg)	2
Drive motor(w)	4 × 50
Wireless LAN protocol	802.11 N
Whether fully compatible with ROS	Yes



(a) A Lizi robot with a robot arm (b) The armless Lizi robo

Fig. 4.23 Lizi™ robot physical photo. (a) Lizi™ robot with mechanical arm; (b) Lizi™ robot without mechanical arm

Table 4.4 Host hardware parameters

Component	Model number
GPU	Nvida titan X
Video memory	16GB
CPU	i7-6700K
Internal storage	32GB

The Lizi™ robot is equipped with sensors such as RGB-D cameras, laser radars, and sonars. The experiments involved in this technical case mainly used its laser radar. The hardware parameters used in the experiment are shown in Tables 4.4 and 4.5.

Table 4.5 Lizi™ robot laser radar parameters

Parameter	Value
Lidar maximum ranging range(m)	4
Lidar angular resolution(°)	0.36
Lidar maximum measurement angle(°)	200
Lidar scan period(ms)	100

4.3.5.3 Technical Method

Selection of Algorithm Model

In the experiments of this technical case, a combination of the classic and efficient A* algorithm and DWA algorithm was used as the control group, in reinforcement learning and deep reinforcement learning (deep reinforcement In the choice of deep reinforcement learning (DRL) methods, as shown in the results of literature retrieval related to robot path navigation, the performance, and effects of using DRL algorithms significantly outperform those of using ordinary reinforcement learning algorithms. Therefore, Q-learning in reinforcement learning is used as the skeleton to build the algorithm on top of it. Among two different types of DRL algorithms, a value-based Deep Q Network (DQN) is used to solve the practical exploration topic. The preference for value-based DRL algorithms over policy-based DRL algorithms mainly depends on the following analysis:

- (1) Policy-based DRL algorithms excel at choosing actions in continuous and large action spaces. In the problem of robot path planning, this characteristic is manifested by sampling the robot’s running speed through a policy-based DRL algorithm and generating trajectories, taking partial derivatives on policy parameters, and completing gradient descent. However, it is necessary to sample the robot’s speed, and the computational resources may be insufficient when the robot is running or simulating motion on the host, which may lead to low sampling frame rate, slow algorithm training speed, etc., easily leading to low training efficiency.
- (2) Policy-based DRL algorithms are very sensitive to parameter selection and are prone to accidental factors caused by human intervention.
- (3) Policy-based DRL algorithms, such as policy gradient algorithms, control the policy of robot path planning, which is essentially a probability density function controlled by parameters. Therefore, the planning model trained by parameter-controlled DRL algorithms such as policy gradient algorithms has a certain randomness when deployed, and the robot path planning problem, as a research topic that involves a high probability of human-machine interaction, hopes that its trained policy model is more stable and less random.
- (4) As a value-based DRL algorithm, the DQN algorithm and the Q-learning algorithm are from the same school, both using a Q matrix to save reward values. The trained model will index the Q matrix for the current state S and greedily choose the action with the highest reward value in the Q matrix to execute. From this perspective, using DQN and other value-based DRLs to train this technical case to solve the problem is more stable and more in line with the expectations of human designers.



Fig. 4.24 In this experiment, the URDF file declares the robot model

Simulation Method Experiment

Whether it is a reinforcement learning method or a deep reinforcement learning method, it is based on the idea of giving the robot continuous “trial and error” opportunities to learn policies. In the work involving robots, this is often difficult to complete in the real world. This is also why robot reinforcement learning method experiments are usually completed in a simulation environment. Most of the experiments in this technical case are implemented in the Gazebo simulation environment under ROS.

To complete the simulation using the Gazebo simulation environment, it is necessary to import the URDF model of the LiziTM robot [30]. URDF is a representation file of the robot model, written in a specially defined markup language. The robot model in this technical case is provided by the official extension package, as shown in Fig. 4.24.

Robot System Control

To achieve real robot control, a desktop computer is needed as the core machine, referred to as the Master machine below, and a router is used to create a local area network. Using the principle of ROS distributed multi-machine communication, the desktop computer acts as the main node, and the robot acts as a distributed node. In the same local area network, the desktop computer can obtain the sensor information of the robot and publish related topics, thereby controlling the robot’s movement.

Configure the router’s IP address to create a local area network, and configure the current local area network’s IP address for both the host and the robot. Add the host’s IP address, the host’s hostname, the robot’s IP address, and the robot’s

hostname to the `/etc/hosts` path (this path may vary depending on the Linux distribution) on the host. Add the command “`export ROS_MASTER_URI = http://djc:11311`” to the console configuration files in both the Master machine and the robot to set the Master machine’s information and the port for transmitting messages. Additionally, add the commands “`export ROS_HOSTNAME = localhost`” and “`export ROS_HOSTNAME = NameOfRobot`” to the console configuration file on the robot. Then, confirm whether the network between the main node and the distributed nodes is unobstructed, and test whether the network environment has been successfully set up.

Start the ROS core on the Master machine to open the communication port, otherwise the robot will not be able to connect to `http://djc:11311`. Connect via remote desktop, remotely connect to the Lizi™ robot from the Master machine, enter the password to access the robot’s terminal, and enter “`$roslaunchlilizizi.launch`” in the robot terminal to start the robot’s launch file and start the robot’s controller and other hardware devices.

DQN Algorithm Completes Path Planning

- (1) Implementation of DQN algorithm: This technical case is based on the ROS system and implements the path planning algorithm for the Lizi™ robot platform based on the DQN algorithm. To correctly configure Lizi™ for use in the ROS environment, the Lizi™ configuration package provided by Robtican officials is required.

The robot will take samples from its laser radar as its observation of the environment, defining six different actions {forward, left turn 90°, left turn 45°, right turn 90°, right turn 45°, stop} as the robot’s output in response to its observed environment. First, initialize the parameters used by the DQN algorithm and the two neural networks required therein, namely, Target-Net and Eval-Net. During the algorithm’s operation, the robot i at time t receives the current environment o_i^t , selects actions according to the ϵ -greedypolicy, and obtains the next state o_i^{t+1} and the currently obtained reward value r from the environment. The state o_i^t , action a , next state o_i^{t+1} , and reward value r are stored as a batch in the memory library. When the number in the memory library reaches 64, 64 batches are randomly selected from it, the current state is assigned to $batch_1$, and Q_{target} is calculated from the next state o_i^{t+1} through Target-Net and assigned to $batch_2$. Here, the assignment should first assign the Q_{eval} obtained from the state o_i^t through Eval-Net to $batch_2$, and then assign Q_{target} to the corresponding action in $batch_2$. Then, the loss function $L_i(\theta_i)$ is calculated based on the mean square error of Q_{eval} and Q_{target} , and only one of the six actions will be adopted by the robot, so this will make the difference of the other five irrelevant actions 0 and not learn. Then, use the loss function value obtained from this batch to perform backpropagation, thereby updating the

network parameters of Eval-Net, and updating Target-Net. After Eval-Net is updated 2000 times, the parameters are fully assigned to Target-Net, and ϵ in the ϵ -greedypolicy is updated after each episode ends, so that the robot's action selection gradually approaches the action provided by the neural network as the network improves.

The ϵ -greedypolicy algorithm mentioned above is a decision-making method for the robot to take actions when training the DQN algorithm. The ϵ -greedypolicy is: each action exploration is randomly selected with a probability of ϵ , and the corresponding action with the maximum Q value from the Q matrix is selected with a probability of $1-\epsilon$. ϵ changes with the iteration number it as shown in formula (4.7):

$$\epsilon_{it} = \begin{cases} \epsilon_0 * (\epsilon_{decay})^{it} & (\epsilon_{it} > \epsilon_{min}) \\ \epsilon_{min} & (\epsilon_{it} \leq \epsilon_{min}) \end{cases} \quad (4.7)$$

where ϵ_0 is the initial value of ϵ , ϵ_{decay} is the decay coefficient of ϵ , its size affects the speed of ϵ decreasing with the update of neural network parameters, ϵ_{min} is the minimum value allowed for ϵ , usually ϵ_{min} is set to a decimal slightly larger than zero.

Finally, suppose the robot's coordinate position is $p_a(x_a, y_a)$, the speed is $\vec{v}(v_x, v_y)$, the shape is approximately a circle with a radius of R , the target point coordinates in this epoch are $p_b(x_b, y_b)$, and the obstacles in the scene are B_k . The reward function r used in the DQN algorithm designed in this technical case is shown in formula (4.8):

$$r = \begin{cases} 200 & (\|p_a - p_b\| < R) \\ -200 & (\Delta t \geq time_{max} \cup \|p_a - B_k\| > R) \\ \frac{|\tan^{-1}(\vec{v}) - \tan^{-1}(\overrightarrow{p_a p_b})| * \phi}{\|p_a - p_b\| * \pi} & \text{otherwise} \end{cases} \quad (4.8)$$

where $time_{max}$ is the maximum time allowed for the robot to find its way, and this technical case sets it to 120 s. $*\phi$ is a parameter to adjust the size of the reward value. As can be seen from the above formula, the robot will be rewarded when it reaches the target point, and it will be punished if it does not reach or collide with objects in the environment after $time_{max}$. In other cases, the size of the reward value will be positively correlated with the angle difference between the speed direction and the target point direction, and inversely related to the position of the robot and the target point.

(2) DQN algorithm parameter settings: The design parameters in the formulas described in the experiments conducted in this technical case are shown in Table 4.6, and the selected neural network structure is shown in Table 4.7.

Table 4.6 Parameters

Parameter identifier	Express meaning	Value
$\text{number}_{\text{states}}$	The number of states that the robot recognizes for the environment	24
$\text{number}_{\text{actions}}$	The number of actions a robot can take in the face of an environment	6
α	Learning rate: Indicates the speed of DQN network convergence	0.0025
γ	The reward and penalty discount indicates the impact of future rewards and penalties on the present	0.99
ε_0	The initial value of ε	1.0
$\varepsilon_{\text{decay}}$	Attenuation coefficient of ε	0.99
ε_{min}	The minimum value allowed by ε	0.08
ϕ	Reward value resizing parameter	2.0

Table 4.7 Neural network structure

Network name	Size of input	Size of output	Activation function
Fc1	$\text{number}_{\text{states}}(24)$	64	Relu
Fc2	64	64	Relu
Dropout	64	64	20%drop
Fc3	64	$\text{number}_{\text{actions}}(6)$	Relu
Activation	$\text{number}_{\text{actions}}(6)$	$\text{number}_{\text{actions}}(6)$	Linear
Optimization algorithm		RMSProp	

4.3.5.4 Results Display

Training of DQN Algorithm

In the Gazebo simulation environment, create a world, import relevant models, add some static obstacles, manually adjust their positions and sizes, and save the model for later use. Create a launch file, import the model file of the Gazebo world, configure related parameters, then use the model generation function in the Gazebo environment to import the URDF model of the robot and initialize the robot's position. After the environment is set up, subscribe to the /odom and /cml_vel topics and start the DQN algorithm module, allowing the robot to learn in the current scene, update network parameters, and reset the scene whenever the robot reaches the target or a collision occurs, and continue learning. After learning is over (3000 episodes in this technical case), use the model.save() function in the Keras package to save the well-learned model. When using it, get the parameters in the file through the model.load_weights() function and assign them to the current network to import the model. The effect of the learning process is shown in Fig. 4.25.

Figure 4.25 is a top-view schematic diagram of the Lizi™ robot training the DQN algorithm in a scene. The brown walls in the scene are the map boundaries and the green square is the robot that is making trial and error and learning in the scene. The red square area represents the robot's target area, which is randomly generated

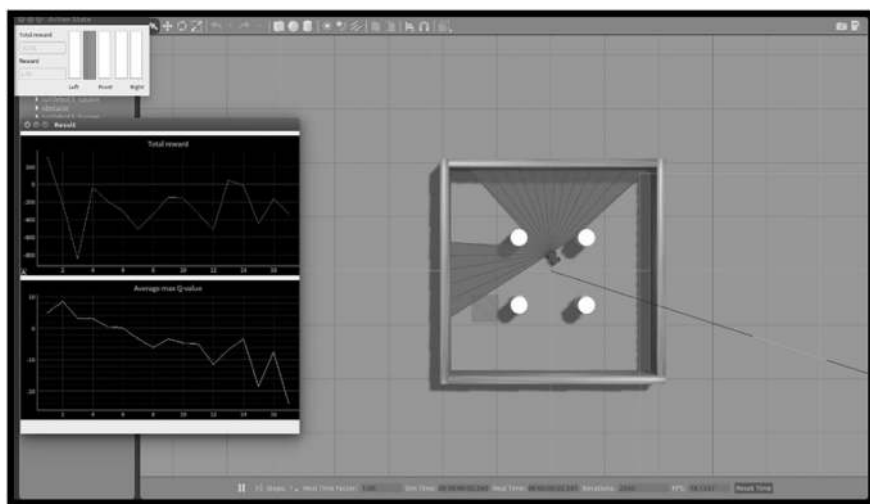


Fig. 4.25 Schematic diagram of training DQN algorithm in Gazebo simulation environment

within the scene map boundary, and the white cylinders are obstacles. In Fig. 4.25, you can visually observe the total reward value during training, the average maximum Q value, and the actions taken by the robot in different environments.

As can be seen from Fig. 4.26, although the total reward received by the robot fluctuates up and down with the progress of training (because it sometimes takes random actions to jump out of the local optimum), the average maximum Q value shows an upward trend. This image also reflects that with the training of the DQN network, the robot gradually learns to take actions corresponding to larger Q values, that is, to reach the target area through a better path. As can be seen from Fig. 4.26, at the end of the algorithm training, the average maximum Q value basically converges from 0 to about 250, which is exactly the reward value set in the algorithm for reaching the destination. This result indicates that, in the later stage of training, each epoch can reach the target point.

Real Robot Implementation

Start the car's controller, sensors, and other hardware devices in the host by calling the launch file. First, based on the traditional methods mentioned above, complete the deployment of the real robot scheme. To achieve this goal, use the gmapping package to build a map in the current indoor environment. After the map is built, import the map into the move_base package, configure the parameters, publish the target point coordinates, and the car can reach the target position. However, in the case of obstacles, the obstacle avoidance is time-consuming and inefficient, and the time spent hovering in front of obstacles is too long. Then implement the reinforcement learning algorithm, first publish the target point coordinates, get the laser data,

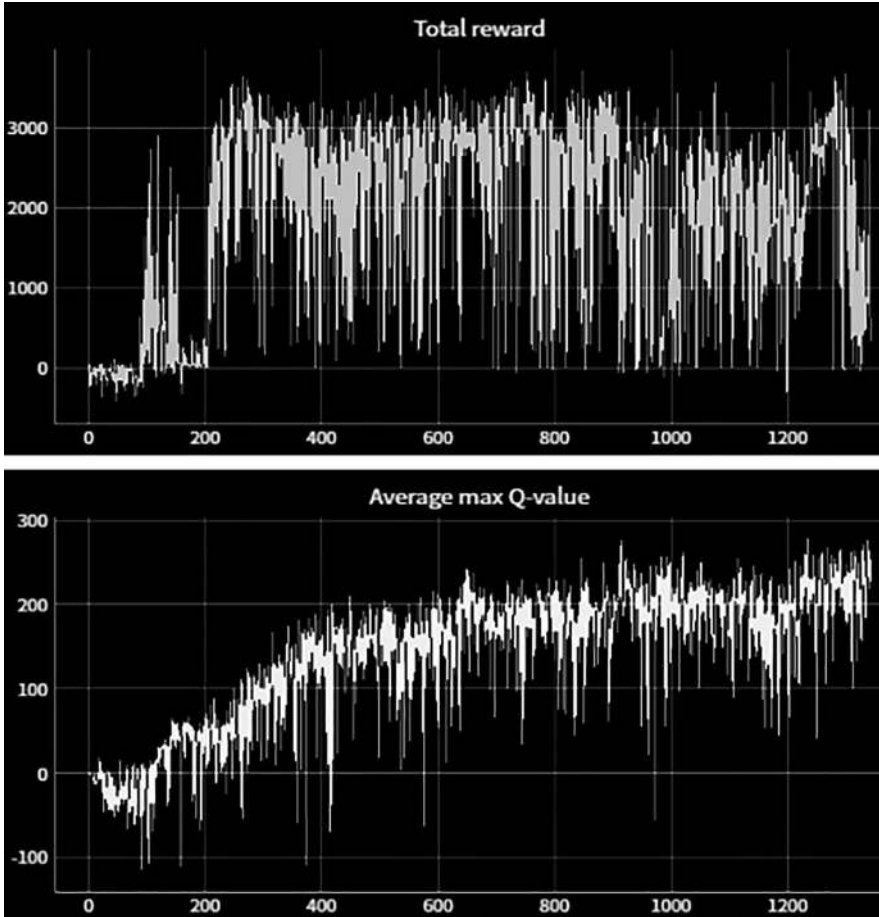


Fig. 4.26 Convergence curve of training DQN algorithm

input it into the saved neural network model, read the output speed of the network, and publish it to the robot, achieving efficient navigation and obstacle avoidance.

4.3.5.5 Summary

This technical case starts from the perspective of theoretical analysis, based on the Lizi™ robot platform in the ROS environment, and first carries out theoretical derivation of traditional methods, reinforcement learning, and DRL methods in robot path planning algorithms. Subsequently, this technical case uses traditional algorithms represented by the A* algorithm and the DWA algorithm as a control group and implements a robot path planning algorithm based on the DQN algorithm in the ROS environment. The designed and implemented algorithms are deployed on the Lizi™ platform for effect evaluation. The results show that the navigation package

provided in ROS has accurate global path planning and can achieve fast and efficient navigation functions, but the obstacle avoidance effect is not ideal. Although obstacle avoidance can be achieved, due to the need to traverse a certain range of speeds, the calculation amount is too large, especially in the real environment, the onboard computer does not have strong computing power, which will cause the obstacle avoidance time to be too long and the efficiency to be too low. The algorithm implemented in this technical case, that is, the robot path planning algorithm under the DQN algorithm, proposes an end-to-end solution, which directly maps the laser data as input to the speed control output (such as forward and left turn), making intelligent and efficient path planning algorithms possible.

4.3.6 Case Study: Monocular Camera-Based Autonomous Obstacle Avoidance for Complex Scenes

4.3.6.1 Task Description

This technical case proposes a vision-based deep reinforcement learning framework to deal with the collision avoidance problem of mobile robots during navigation. This framework relies only on an RGB monocular camera and consists of two modules: an environment perception module and a control decision module. In the environment perception module, the RGB data obtained by the monocular camera deployed on the mobile robot is encoded to obtain one-dimensional pseudo-laser data. In the control decision module, a model based on deep reinforcement learning is used, which accepts the pseudo-laser data obtained by the environment perception module, and then publishes collision avoidance control decisions.

The main tasks of this work are as follows:

- (1) Propose a new vision-based robot navigation obstacle avoidance method that relies solely on an RGB monocular camera as a sensor.
- (2) In order to take advantage of laser sensors and explore the semantic information of the environment, it is proposed to use RGB data to generate one-dimensional pseudo-laser observations. At the same time, a new depth map slicing method is proposed, which greatly improves the ability of the intelligent agent to avoid irregular objects.
- (3) In order to adapt to the proposed vision-based model, a new data enhancement function is introduced to improve the stability of the model and reduce the gap from the simulation environment to the real environment.



Chapter Knowledge Points

Encoder: A device that encodes and converts signals or data into a signal form that can be used for communication, transmission, and storage.

Decoder: A hardware/software device that can decode and restore digital audio and video data streams into analog audio and video signals.

Neural Networks: A mathematical model of an algorithm that mimics the behavioral characteristics of animal neural networks and performs distributed parallel information processing. This network relies on the complexity of the system and adjusts the relationship between the numerous internal nodes to achieve the purpose of processing information.

Depth Prediction: A computer vision task aimed at estimating depth from 2D images. This task requires inputting an RGB image and outputting a depth image, including information about the distance from the viewpoint to the object in the image.

NVIDIA: NVIDIA is a global leader in programmable graphics processing technology, focusing on creating products that enhance the human-computer interaction experience of personal and professional computing platforms.

4.3.6.2 Related Work

Fully Convolutional Network

Fully convolutional networks (FCN) achieve pixel-level regression on images, thereby solving pixel-level depth prediction and image segmentation problems. The FCN upsamples the feature map of the last convolutional layer of the downsampling module to restore it to the same size as the input image, thus generating a prediction for each pixel and accepting input images of any size. As shown in Fig. 4.27, the FCN finally uses a SoftMax classifier to perform pixel-level classification on the feature map, thereby obtaining the final segmentation map.

In summary, the fully convolutional network mainly includes an encoder and a decoder. This technical case uses the efficient convolutional neural network model (MobileNet) [59] for mobile vision as the encoder part for depth prediction and image segmentation to reduce the amount of parameters; uses linear interpolation as the upsampling method; uses skip connections to retain the texture color information of the image; and uses NetAdapt as the network optimizer to embed it into the mobile robot platform.

- (1) MobileNet network structure: Depthwise separable convolution network performs bitwise multiplication in the channel, at which time the number of channels does not change; then the result of the first step is convolved with a 1×1 convolution kernel in the traditional way, changing the number of channels of the feature, and its calculation method is defined as follows:

$$\hat{G}_{k,l,n} = \sum_{i,j} \hat{K}_{i,j,m,n} \bullet F_{k+i-1,l+j-1,m} \quad (4.9)$$

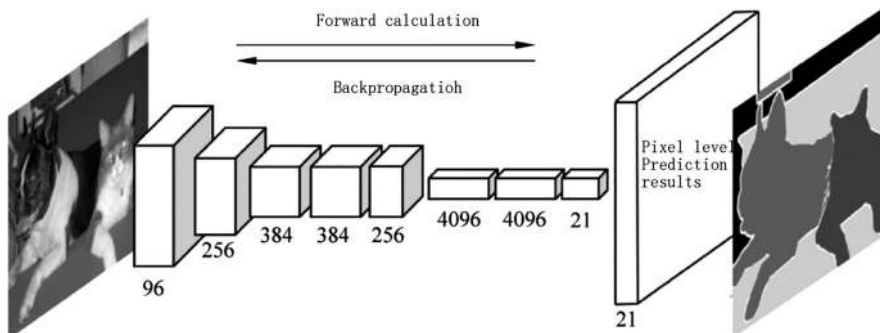


Fig. 4.27 Structure diagram of the fully convolutional network

The calculation method of the depthwise separable computation amount is as follows:

$$D_K \times D_K \times M \times D_F + 1 \times 1 \times M \times N \times D_F \times D_F \quad (4.10)$$

where D_F is the size of the feature map, D_K is the size of the convolution kernel, M is the number of channels of the input feature, and N is the number of channels of the output feature.

Experiments have shown that using MobileNet as the encoder of FCN reduces the number of network parameters and realizes the real-time operation of depth prediction algorithms and image segmentation algorithms on mobile robot-embedded devices.

- (2) Nearest neighbor interpolation: In order to improve the running efficiency of FCN in the decoder stage, improve the running speed of the depth prediction algorithm and image segmentation algorithm, and meet the real-time requirements of the experiment, this technical case uses nearest neighbor interpolation as the upsampling method in the decoder stage. Nearest neighbor interpolation is the simplest interpolation algorithm, which does not require calculation, and directly assigns the value of the pixel closest to the pixel to be calculated to the pixel to be calculated. For example, suppose the size of the source image A is $m \times n$, and the size of the image B after nearest neighbor interpolation is $a \times b$, from which the scaling factor can be obtained as follows:

$$K = \frac{a}{m} = \frac{b}{n} \quad (4.11)$$

Now take a point in image B, you can know the theoretical calculation position in image A is

$$\left(\frac{x_0}{K}, \frac{y_0}{K} \right) \quad (4.12)$$

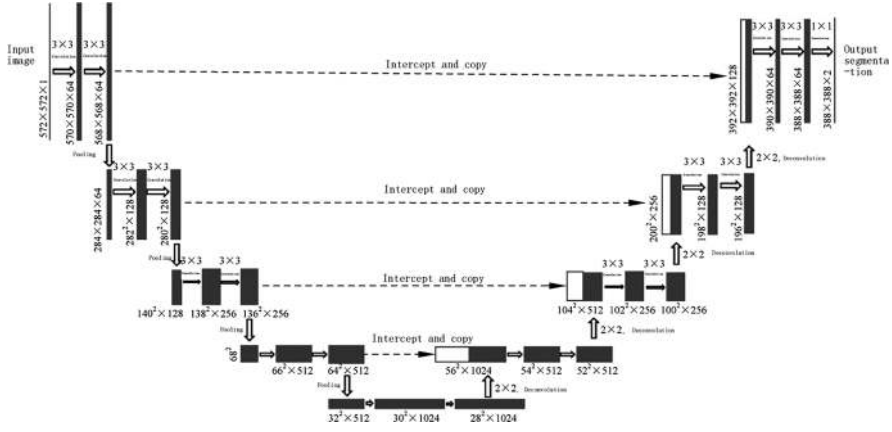


Fig. 4.28 U-Net network structure

The theoretical position value obtained in A may be a decimal, which means that this point has no actual corresponding point in A. At this time, the obtained position should be rounded, that is, the point in the image closest to this theoretical point is taken as its value. This interpolation method is called nearest neighbor interpolation.

- (3) Skip connections: Depth prediction and image segmentation are both pixel-level regression tasks, so U-Net is usually used as the network architecture method, because U-Net is a typical encoder-decoder network structure (as shown in Fig. 4.28).

In the encoder network, many downsampling layers (convolutional layers and pooling layers) are typically used to reduce the spatial resolution of the input image and obtain higher-dimensional abstract features. However, it is difficult to recover the lost image details during the upsampling process, which makes it difficult for the decoder network structure to accurately predict a high-resolution result image. In order to obtain accurate depth maps and semantic maps, this technical case adopts the method of skip connections. Skip connections allow some details in the image to be directly mapped from the high-resolution feature map to the feature map in the decoder network structure, which enables the decoder network structure to reconstruct a high-resolution output full of details.

Skip connections used in ResNet and DenseNet can effectively reduce the problems of gradient vanishing and network degradation, making the deep network easier to train. For neural networks, as the network structure deepens, the network's expressive power will become stronger and the performance will become better. However, as the network structure deepens, it will also bring various problems, such as gradient vanishing and gradient explosion. The residual block represents the network output as a linear superposition of the input and a nonlinear transformation of the input, breaking the symmetry of the network, making the deep neural network easier to train, and its calculation method is defined as follows:

$$y = F(x, \{w_i\}) + x, \quad (4.13)$$

where x and y are the input and output of the network, respectively, and $F(x, \{w_i\})$ represents the mapping that the residual block needs to learn.

Partially Observable Markov Decision Process

POMDP is a general Markov decision process, which simulates the decision-making process of a mobile robot assuming that the system dynamics are determined by the Markov decision process (MDP), but the mobile robot cannot directly observe its state during the decision-making process. Instead, it must infer the distribution of states based on the results of global and partial area observations of the model. Formally, POMDP is the following seven-tuple:

$$(S, A, T, R, \Omega, O, \gamma) \quad (4.14)$$

where S is a set of states, A is a set of actions, T is a set of conditional transition probabilities between states, $R: S \times A \rightarrow R$ is a reward function, Ω is a set of observations, O is a set of conditional observation probabilities, and $\gamma \in [0, 1]$ is a discount factor. At each time period, the mobile robot is in a certain state $s \in S$, the mobile robot will take some action $a \in A$, causing the mobile robot to transition from state s to state s' , with a state transition probability of $T(s'|s, a)$. Similarly, the mobile robot receives an observation value of $o \in O$, which depends on the new state of the environment, with a probability of $O(o|s', a)$. Finally, the mobile robot receives a reward r equal to $R(s, a)$, and the process is repeated. The goal of the entire process is to allow the mobile robot to anticipate the action that maximizes its future discounted reward at each time

$$E \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (4.15)$$

where the discount factor γ determines how much direct reward there is for longer-term rewards. When $\gamma = 0$, it means that the mobile robot only cares about which action will produce the largest expected immediate reward. When $\gamma = 1$, it means that the mobile robot cares about maximizing the expected total future reward.

In order to solve the challenges brought by the partially observable Markov decision process, this technical case introduces the Long Short-Term Memory Recurrent Neural Network and Attention Mechanism Network to solve the problem of a limited field of view. At the same time, this technical case uses the reinforcement learning framework and the actor-critic framework to solve the obstacle avoidance problem of the mobile robot.

- (1) Long Short-Term Memory Recurrent Neural Network: Due to the limitations of the sensor device's field of view, mobile robots cannot fully acquire the sur-

rounding state, making the training process of mobile robots a partially observable Markov decision process. To compensate for the limited field of view of mobile robots during the training process, a feasible solution is to endow mobile robots with memory capabilities, allowing them to model their current state based on the memory of past states and the current observed state, thereby better understanding the surrounding environment.

This technical case uses a recurrent neural network (RNN) to endow mobile robots with memory capabilities. RNN is a type of neural network specifically designed to handle sequence data. Compared to traditional neural networks, RNN can handle data with varying sequence lengths, and by continuously cycling information, it ensures the persistence of information to effectively solve problems containing temporal information. However, RNN has a long-term dependency problem. Since the training process of mobile robots is also a sequential process, and the exploration sequence is long, using RNN cannot aggregate memory states far from the current observed state, which is not conducive to the mobile robot's understanding of the current environment. To ensure that mobile robots can aggregate all states of the current exploration sequence during the exploration process, a long short-term memory network (LSTM) is used. LSTM is a special type of recurrent neural network, as shown in Fig. 4.29. The goal of this network design is to solve the long-term dependency problem in RNN. In standard recurrent neural networks, the repeating module has a very simple network structure, with only a single tanh layer. Unlike standard RNNs, LSTM also has this chain structure, but its repeating unit is different from the single layer in the repeating unit of the standard RNN, and it has four network layers inside.

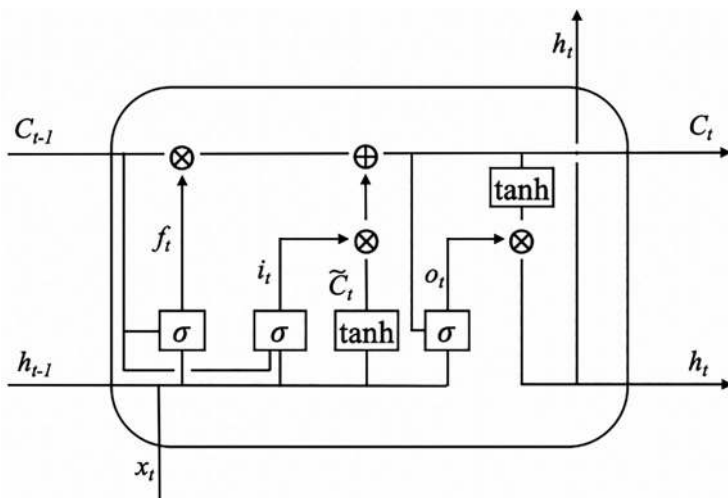


Fig. 4.29 LSTM network structure

(2) Actor-Critic Reinforcement Learning Framework: This technical case uses the actor-critic (AC) framework based on policy gradient to implement the reinforcement learning obstacle avoidance strategy. The actor-critic framework is widely used in practical reinforcement learning algorithms. This algorithm integrates value function estimation and policy search algorithms and is the most commonly used reinforcement learning framework in solving practical problems.

The actor-critic algorithm originated from the policy search algorithm, which is an important branch in the Markov decision process, as shown in Fig. 4.30. The policy search algorithm parameterizes the obstacle avoidance strategy of the mobile robot, for example, using linear or nonlinear (such as neural network) functions to represent the strategy, and then finds the optimal parameters that maximize the expected cumulative return through policy optimization.

To improve the convergence speed of the algorithm, the proximal policy optimization algorithm (PPO) based on the actor-critic framework is used. Compared with traditional policy gradient algorithms, PPO uses importance sampling technology to achieve the reuse of training data, without the need to sample training data after each parameter update. At the same time, to constrain the distance between the sampling policy and the training policy, PPO introduces KL divergence into the return function, reducing the difficulty of training the reinforcement learning algorithm.

4.3.6.3 Technical Method

The research content of this technical method mainly involves two parts: the environmental perception module and control decision module, as shown in Fig. 4.31. The environmental perception module mainly involves the network structure design of depth prediction and semantic segmentation, and the operation mode of local dynamic minimum pooling. The control decision module mainly involves the

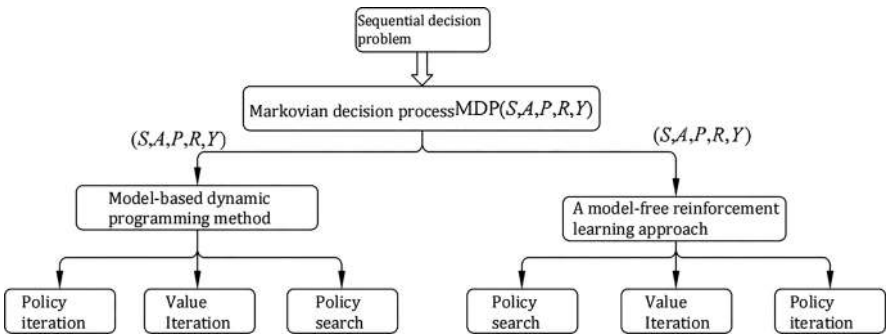


Fig. 4.30 Classification of Markov decision processes

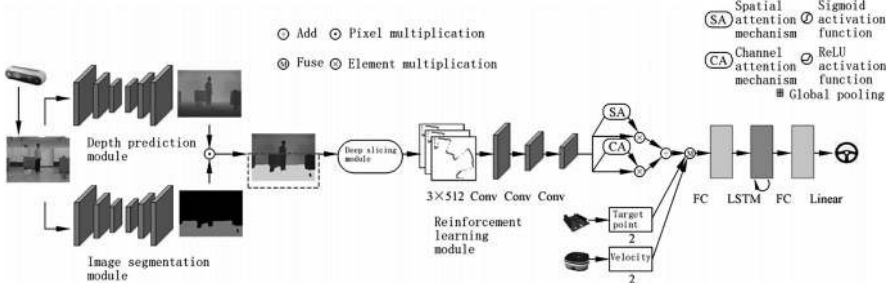


Fig. 4.31 Network framework

impact of the introduction of attention mechanism and LSTM mechanism on the obstacle avoidance strategy of deep reinforcement learning.

Environmental Perception Module

The environmental perception module encodes the RGB data captured by the mobile robot to obtain one-dimensional pseudo-laser data containing object contour information. In order to obtain one-dimensional pseudo-laser data, depth prediction, and image processing algorithms are used to obtain the corresponding depth map and 0–1 mask map, respectively. Then, the mask map is used to remove useless information in the depth map to obtain a depth map without ground distance information. Finally, the local dynamic minimum pooling operation proposed in this technical case is used to slice the depth map without ground distance information to obtain the final one-dimensional pseudo-laser data. The specific operation is shown in Fig. 4.32.

- (1) **Depth Prediction Module:** Depth prediction is an important function for mobile robots and has wide application value in tasks such as robot mapping, localization, and obstacle detection. The existing depth sensors mostly have the disadvantages of high price, high energy consumption, and large volume, such as lasers and structured light sensors. These disadvantages make it difficult for existing depth sensors to be deployed and used on small robot platforms, which also promotes the development of predicting depth maps with a single RGB data, because this method has the advantages of low price, low energy consumption, and small volume.
- (2) **Image Segmentation Module:** The so-called image segmentation refers to the image segmentation algorithm dividing the image into several complementary but nonintersecting regions based on the performance of the image in pixel color, grayscale, geometric shape, texture, and other features. The image segmentation algorithm makes the texture, color, and other features in the same area show consistency or similarity, and show a huge gap in different areas.

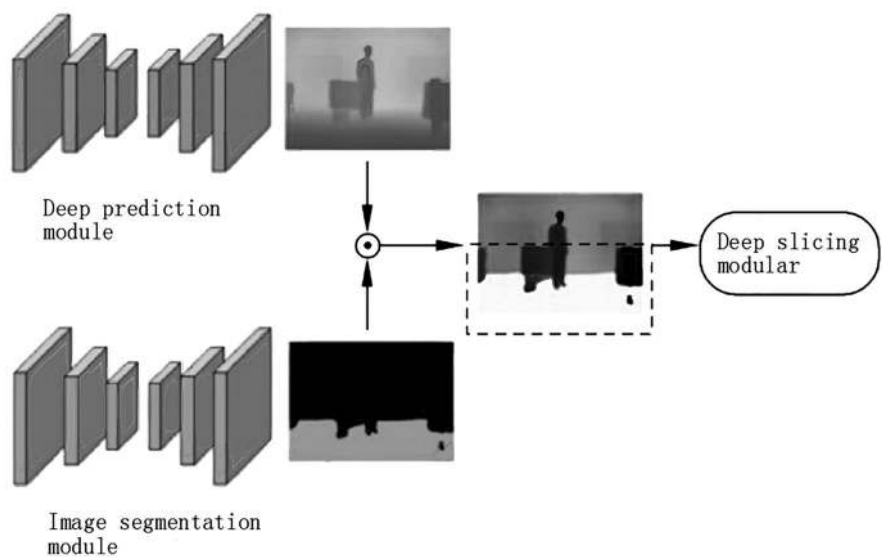


Fig. 4.32 Environmental perception module

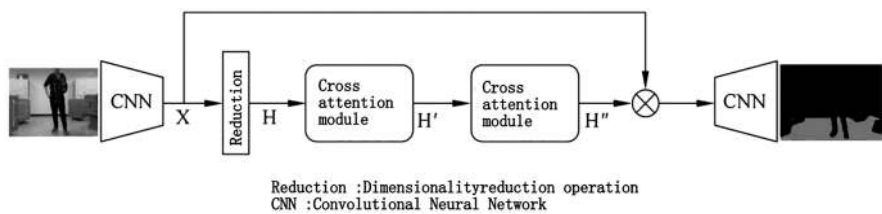


Fig. 4.33 Image segmentation network structure

The image segmentation network structure is shown in Fig. 4.33. First, the image goes through a convolutional neural network to produce a feature map X , with a spatial size of $H \times W$ (height, width). In order to retain more details (texture, color) and efficiently generate dense feature maps, the last two downsampling operations are removed and replaced with dilated convolution. This method makes the H and W of the output feature map X 1/8 of the original input image. After obtaining the corresponding feature map X , the feature map X is further dimensionally reduced by the convolutional layer to obtain H . Then, in order to aggregate contextual information, the feature map H is input into the cross-attention module to generate a new feature map H' , so that each pixel can capture contextual information in a crisscross manner. However, H' only aggregates the contextual information in the horizontal and vertical directions. In order to obtain richer and denser contextual information, H' is input into the cross-attention module again, and the feature map H'' is output. In this way, each pixel in the feature map H'' aggregates pixel information from any other position. In order to reduce the number of parameters, the two cross-attention

modules share the same parameters to avoid adding too many extra parameters and slowing down the calculation speed. In order to obtain the final segmentation map, the feature map H'' , which has aggregated dense contextual features, is connected with the local feature map X , and several convolutional layers and activation layers are used to fuse its features until the final result is output.

- (3) **Data Augmentation:** The training process of reinforcement learning algorithms is a process in which the mobile robot continuously interacts with the environment, constantly collecting training data from its interactions with the environment, and then using the collected training data to update the reinforcement learning strategy. In order to reduce the cost of training deep reinforcement learning algorithms and improve the efficiency of algorithm training, reinforcement learning algorithms need to be trained in a simulated environment first and then transferred to a real environment.

However, there is a huge gap between the simulated environment and the real environment, such as the color and texture of objects, so it is difficult for the algorithm trained in the simulated environment to be transferred to the real environment. In order to narrow the gap between the data collected in the simulated environment and the real environment, in addition to optimizing the performance of the algorithm to improve the accuracy of the data collected in the real environment, another way is to add noise to the data collected in the simulated environment, making the data collected in the simulated environment closer to the data collected in the real environment. Using the noisy laser data to train the reinforcement learning obstacle avoidance strategy can improve the robustness of the reinforcement learning algorithm to noise and improve the effect of transferring the reinforcement learning algorithm from simulation to reality.

Control Decision Module

The control decision module mainly receives the pseudo-laser data processed by the environmental perception module, the position of the target point, and the speed of the mobile robot at the previous moment, and then regresses the correct obstacle avoidance behavior that should be taken at the current moment. In order to help the mobile robot better grasp and handle the received environmental data in the control decision module, this technical case introduces the LSTM mechanism and attention mechanism in the control decision module, the specific structure is shown in Fig. 4.34:

- (1) **Recurrent Neural Network Mechanism:** By adding LSTM to the model to introduce temporal information, the mobile robot is given memory capability. Due to the influence of the field of view, it is difficult for the mobile intelligent agent to make accurate judgments about the current state. In order to improve the ability of the mobile intelligent agent to understand the surrounding environ-

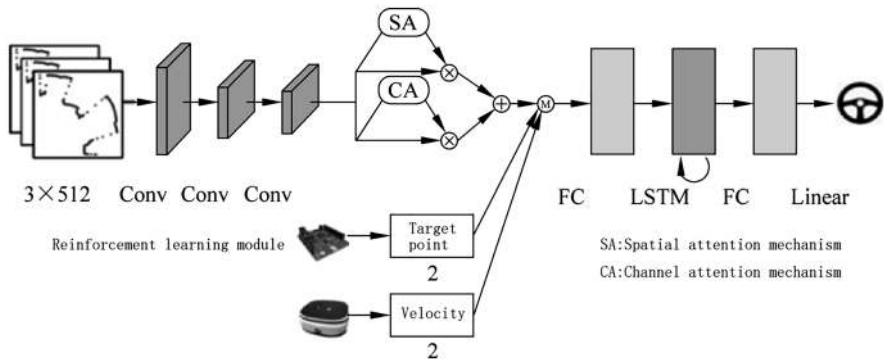


Fig. 4.34 Control decision module

ment, LSTM is introduced. Because the training of the reinforcement learning obstacle avoidance strategy is a continuous interaction between the mobile robot and the environment, collecting experience data, and using the experience data for training, that is, the mobile robot starts from a specific state and continues until the end of the task, which is called a complete Episode, its mathematical representation is as follows:

$$\tau = \{s_1, a_1, s_2, a_2, \dots, s_T, a_T\} \quad (4.16)$$

Therefore, in the process of interacting with the environment, there is a temporal relationship between each state in a sequence. This technical case introduces LSTM to enable the mobile robot to remember past states in the current sequence, grasp the relationship between states, so that the mobile robot can better understand its current state based on the memory of past states, and make up for the defects in the sensor's field of view.

- (2) **Attention Mechanism Module:** This technical case introduces the attention mechanism, allowing the mobile robot to more effectively focus on the objects it needs to pay attention to. As a powerful deep learning tool, the attention mechanism is embedded as an important module in various models to handle a variety of tasks. The attention mechanism mimics the internal process of biological observation behavior, that is, a mechanism that aligns internal experience and external sensations to increase the observation precision of certain areas. For example, when a person observes an image, they will quickly scan the entire image, then find the target area that needs to be focused on, which is the focus of human attention. After finding the target area, humans will invest more attention resources in this target area to obtain more detailed information about the target area that needs attention, and reduce attention to other useless information.

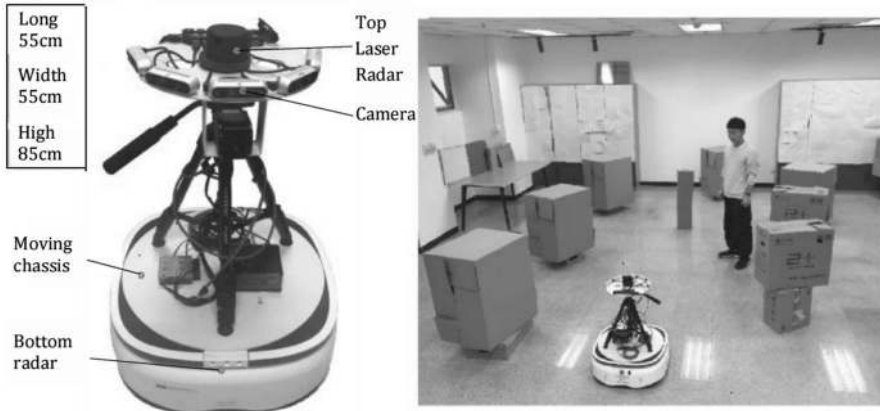


Fig. 4.35 Real machine test equipment and site

4.3.6.4 Results Display

To check the overall performance of the method proposed in this technical case in the real world, a ROS-based mobile robot as shown in Fig. 4.35 was deployed in a real scene composed of cardboard boxes, tables, and chairs. The mobile robot can transmit the acquired images to a local host equipped with an Nvidia GTX1080 GPU for image processing and action prediction through its own router, and then send the speed information to the robot to guide the mobile robot to avoid obstacles. The calculation speed can reach 10 Hz, meeting the real-time requirements of the experiment. The experiments in real scenes have shown that the method proposed in this technical case can effectively avoid dynamic and static obstacles in the scene, and has good robustness and generalization for static irregular objects and challenging dynamic pedestrians.

4.3.6.5 Summary

Mobile robot obstacle avoidance technology, as a basic function of mobile robots, has a wide range of research and application value. In recent years, with the development of deep learning, learning-based obstacle avoidance algorithms have gradually replaced traditional obstacle avoidance algorithms. The obstacle avoidance algorithm based on deep reinforcement learning solves the problems of large computation, network delay, and localization failure brought by the central control obstacle avoidance algorithm by giving each mobile robot independent obstacle avoidance ability. However, the obstacle avoidance algorithm based on deep reinforcement learning still has problems of difficult training and difficult transfer. Because there is a huge gap between the simulation environment and the real environment, using vision will not only introduce a lot of redundant information leading to the failure of the reinforcement learning algorithm training, but also cause the

obstacle avoidance algorithm transfer failure due to the huge gap between the simulation environment and the real environment. To solve the previous problems, most researchers use lasers as sensors to train reinforcement learning obstacle avoidance algorithms, because simple and accurate laser data not only includes less redundant information to speed up the convergence of the obstacle avoidance algorithm but also can ignore the huge gap between the simulation environment and the real environment, realizing the effective transfer of the algorithm.

The visual obstacle avoidance algorithm proposed in this technical case promotes the application value of visual obstacle avoidance in real scenes, but there are still some problems that have not been solved. For example, the problem of depth prediction algorithm failure at close range; the problem of image segmentation algorithm failure caused by the ground being made of highly reflective materials, etc. In future research, researchers need to continuously improve the performance of related image processing algorithms, provide high-simulation virtual environments, and achieve end-to-end training from image to action.

4.4 Summary

This technical case introduces tasks related to robot navigation. In terms of localization, it focuses on the introduction of classic camera relocalization and deep learning-based camera relocalization methods, introduces the concept of localization and its subtasks and their relationships, and also introduces the basic concepts involved in localization solutions, including bag-of-words methods, random sample consensus algorithms, etc. The acquisition of camera pose is an important task for the robot to understand its own pose state. After successfully obtaining localization information, when facing actual application scenarios, robot navigation and obstacle avoidance tasks are the basic tasks for robot operation, so they are basic application tasks. Then the technologies related to robot navigation and obstacle avoidance tasks are introduced and explained, including path planning algorithms, classic methods of robot obstacle avoidance, and methods based on deep learning.

4.5 Discussion

1. What are the commonly used feature similarity measurement methods? Please try to implement these measurement methods using Python and analyze their performance differences.
2. Please draw a flowchart for the random sampling consensus method.
3. The relocalization methods introduced in this technical case are all based on static scenes. If it is a dynamic scene with moving objects, will the performance of relocalization decrease? Why? How to solve it?

4. Please briefly describe the implementation process of the reinforcement learning algorithm in obstacle avoidance navigation tasks, and design an efficient reinforcement learning reward function.
5. Briefly describe the process of the A* algorithm and the Dijkstra algorithm, and analyze the advantages and disadvantages and applicable scenarios of the two algorithms.

References

1. BRACHMANN E, ROTHER C. Learning less is more – 6d camera localization via 3d surface regression; proceedings of the 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, F Jun 18–23, 2018 [C]. Ieee: NEW YORK, 2018.
2. BRACHMANN E, ROTHER C. Visual camera re-localization from rgb and rgb-d images using dsac [J]. IEEE transactions on pattern analysis and machine intelligence, 2021, 44(9): 5847–5865.
3. BRACHMANN E, KRULL A, NOWOZIN S, et al. Dsac-differentiable ransac for camera localization; proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, F Jul 21–26, 2017 [C]. Ieee: NEW YORK, 2017.
4. GAO X. Visual SLAM fourteen lectures: from theory to practice [M]. Beijing: Electronic Industry Press, 2017.
5. SHOTTON J, GLOCKER B, ZACH C, et al. Scene coordinate regression forests for camera relocalization in rgb-d images; proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, F Jun 23–28, 2013 [C]. Ieee: NEW YORK, 2013.
6. VALENTIN J, NIESSNER M, SHOTTON J, et al. Exploiting uncertainty in regression forests for accurate camera relocalization; proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, F Jun 07–12, 2015 [C]. Ieee: NEW YORK, 2015.
7. MASSICETI D, KRULL A, BRACHMANN E, et al. Random forests versus neural networks – what’s best for camera localization? [J]. 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017: 5118–5125.
8. CAVALLARI T, GOLODETZ S, LORD N A, et al. On-the-fly adaptation of regression forests for online camera relocalisation; proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, F Jul 21–26, 2017 [C]. Ieee: NEW YORK, 2017.
9. KENDALL A, GRIMES M, CIPOLLA R, et al. PoseNet: A convolutional network for real-time 6-dof camera relocalization; proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, F Dec 11–18, 2015 [C]. Ieee: NEW YORK, 2015.
10. ZHOU L, LUO Z, SHEN T, et al. Kfnet: Learning temporal camera relocalization using kalman filtering; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, F Jun 14–19, 2020 [C]. Ieee: NEW YORK, 2020.
11. TAIRA H, OKUTOMI M, SATTLER T, et al. Inloc: Indoor visual localization with dense matching and view synthesis [J]. Ieee Transactions on Pattern Analysis and Machine Intelligence, 2021, 43(4): 1293–1307.
12. GERMAIN H, LEPETIT V, BOURMAUD G, et al. Neural reprojection error: Merging feature learning and camera pose estimation; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, F Jun 19–25, 2021 [C]. IEEE Computer Soc: LOS ALAMITOS, 2021.

13. HUANG Z, ZHOU H, LI Y, et al. Vs-net: Voting with segmentation for visual localization; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, F Jun 19–25, 2021 [C]. IEEE Computer Soc: LOS ALAMITOS, 2021.
14. SARLIN P E, UNAGAR A, LARSSON M, et al. Back to the feature: Learning robust camera localization from pixels to pose; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, F Jun 19–25, 2021 [C]. IEEE Computer Soc: LOS ALAMITOS, 2021.
15. TANG S, TANG C, HUANG R, et al. Learning camera localization via dense scene matching; proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Electr Network, F Jun 19–25, 2021 [C]. IEEE Computer Soc: LOS ALAMITOS, 2021.
16. BRACHMANN E, ROTHER C. Neural-guided ransac: Learning where to sample model hypotheses [J]. IEEE I Conf Comp Vis, 2019: 4321–4330.
17. RUBLEE E, RABAUD V, KONOLIGE K, et al. Orb: An efficient alternative to sift or surf; proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, F Nov 06–13, 2011 [C]. IEEE: NEW YORK, 2011.
18. BAY H, ESS A, TUYTELAARS T, et al. Speeded-up robust features (surf) [J]. Comput Vis Image Underst, 2008, 110(3): 346–359.
19. LOWE D G. Distinctive image features from scale-invariant keypoints [J]. Int J Comput Vis, 2004, 60(2): 91–110.
20. FISCHLER M A, BOLLES R C. Random sample consensus—a paradigm for model-fitting with applications to image-analysis and automated cartography [J]. Commun ACM, 1981, 24(6): 381–395.
21. JEGOU H, DOUZE M, SCHMID C, et al. Aggregating local descriptors into a compact image representation; proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, F Jun 13–18, 2010 [C]. IEEE Computer Soc: LOS ALAMITOS, 2010.
22. ARANDJELOVIC R, GRONAT P, TORII A, et al. Netvlad: Cnn architecture for weakly supervised place recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(6): 1437–1451.
23. LLOYD S P. Least-squares quantization in pcm [J]. IEEE Trans Inf Theory, 1982, 28(2): 129–137.
24. SIVIC J, ZISSERMAN A, IEEE COMPUTER S, et al. Video google: A text retrieval approach to object matching in videos; proceedings of the 9th IEEE International Conference on Computer Vision, Nice, France, F Oct 13–16, 2003 [C]. IEEE Computer Soc: LOS ALAMITOS, 2003.
25. ROBERTSON S. Understanding inverse document frequency: On theoretical arguments for idf [J]. J Doc, 2004, 60(5): 503–520.
26. GALVEZ-LOPEZ D, TARDOS J D. Bags of binary words for fast place recognition in image sequences [J]. IEEE Trans Robot, 2012, 28(5): 1188–1197.
27. MUR-ARTAL R, MONTIEL J M M, TARDOS J D. Orb-slam: A versatile and accurate monocular slam system [J]. IEEE Trans Robot, 2015, 31(5): 1147–1163.
28. LIU C, YUEN J, TORRALBA A, et al. Sift flow: Dense correspondence across different scenes; proceedings of the 10th European Conference on Computer Vision (ECCV 2008), Marseille, France, F Oct 12–18, 2008 [C]. Springer-Verlag Berlin: BERLIN, 2008.
29. GALVEZ-LOPEZ D, TARDOS J D, IEEE. Real-time loop detection with bags of binary words; proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, Sep 25–30, 2011 [C]. IEEE: NEW YORK, 2011.
30. GEE A P, MAYOL-CUEVAS W. 6d relocalisation for rgbd cameras using synthetic view regression; proceedings of the 23rd British Machine Vision Conference, University of Surrey, Guildford, England, Sep 03–07, 2012 [C]. BMVA Press: GUILDFORD, 2012.
31. GLOCKER B, SHOTTON J, CRIMINISI A, et al. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding [J]. IEEE Trans Vis Comput Graph, 2015, 21(5): 571–583.

32. SCHINDLER G, BROWN M, SZELISKI R, et al. City-scale location recognition; proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, Jun 17–22, 2007 [C]. IEEE: NEW YORK, 2007.
33. CAO S, SNAVELY N. Graph-based discriminative learning for location recognition [J]. *Int J Comput Vis*, 2015, 112(2): 239–254.
34. LI Y P, SNAVELY N, HUTTENLOCHER D P. Location recognition using prioritized feature matching; proceedings of the 11th European Conference on Computer Vision, Heraklion, Greece, Sep 05–11, 2010 [C]. Springer-Verlag Berlin: BERLIN, 2010.
35. LIM H, SINHA S N, COHEN M F, et al. Real-time image-based 6-dof localization in large-scale environments; proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, Jun 16–21, 2012 [C]. IEEE: NEW YORK, 2012.
36. SATTTLER T, HAVLENA M, SCHINDLER K, et al. Large-scale location recognition and the geometric burstiness problem; proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, Jun 27–30, 2016 [C]. IEEE: NEW YORK, 2016.
37. LI Y P, SNAVELY N, HUTTENLOCHER D, et al. Worldwide pose estimation using 3D point clouds; proceedings of the 12th European Conference on Computer Vision (ECCV), Florence, Italy, Oct 07–13, 2012 [C]. Springer-Verlag Berlin: BERLIN, 2012.
38. SATTTLER T, HAVLENA M, RADENOVIC F, et al. Hyperpoints and fine vocabularies for large-scale location recognition; proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, Dec 11–18, 2015 [C]. IEEE: NEW YORK, 2015.
39. SATTTLER T, LEIBE B, KOBBELT L. Efficient & effective prioritized matching for large-scale image-based localization [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, 39(9): 1744–1756.
40. GUZMAN-RIVERA A, KOHLI P, GLOCKER B, et al. Multi-output learning for camera relocation; proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, Jun 23–28, 2014 [C]. IEEE: NEW YORK, 2014.
41. BRACHMANN E, MICHEL F, KRULL A, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image; proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, Jun 27–30, 2016 [C]. IEEE: NEW YORK, 2016.
42. LI X T, YLIOINAS J, KANNALA J. Full-frame scene coordinate regression for image-based localization; proceedings of the 14th Conference on Robotics-Science and Systems, Carnegie Mellon Univ, Pittsburgh, PA, USA, Jun 26–30, 2018 [C]. MIT Press: CAMBRIDGE, 2018.
43. LI X T, YLIOINAS J, VERBEEK J, et al. Scene coordinate regression with angle-based reprojection loss for camera relocation; proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, Germany, F Sep 08–14, 2018 [C]. Springer International Publishing Ag: CHAM, 2019.
44. PFEIFFER M, SCHAEUBLE M, NIETO J, et al. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots [J]. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017: 1527–33.
45. TAI L, LI S H, LIU M. A deep-network solution towards model-less obstacle avoidance [J]. *2016 IEEE International Conference on Intelligent Robots and Systems (IROS 2016)*, 2016: 2759–2764.
46. LONG P X, FAN T X, LIAO X Y, et al. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning; proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, F May 21–25, 2018 [C]. IEEE Computer Soc: LOS ALAMITOS, 2018.
47. CHOI J, PARK K, KIM M, et al. Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view; proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, F May 20–24, 2019 [C]. IEEE: NEW YORK, 2019.
48. SOUHILA K, KARIM A. Optical flow based robot obstacle avoidance [J]. *Int J Adv Robot Syst (Austria)*, 2007, 4(1): 13–6.

49. BILLS C, CHEN J, SAXENA A. Autonomous mav flight in indoor environments using single image perspective cues [J]. 2011 IEEE International Conference on Robotics and Automation (ICRA 2011), 2011: 5776–5783.
50. KIM D K, CHEN T. Deep neural network for real-time autonomous indoor navigation [J/OL] 2015, <https://doi.org/10.48550/arXiv.1511.04668>.
51. LIULONG M, YANJIE L, JIAO C, et al. Learning to navigate in indoor environments: From memorizing to reasoning [J/OL] 2019, <https://doi.org/10.48550/arXiv.1904.06933>.
52. BHARADHWAJ H, WANG Z H, BENGIO Y, et al. A data-efficient framework for training and sim-to-real transfer of navigation policies; proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, F May 20–24, 2019 [C]. IEEE: NEW YORK, 2019.
53. GORDON D, KADIAN A, PARIKH D, et al. Splitnet: Sim2sim and task2task transfer for embodied visual navigation; proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, South Korea, F Oct 27–Nov 02, 2019 [C]. IEEE Computer Soc: LOS ALAMITOS, 2019.
54. SAVVA M, KADIAN A, MAKSYMETS O, et al. Habitat: A platform for embodied ai research; proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, South Korea, F Oct 27–Nov 02, 2019 [C]. IEEE: NEW YORK, 2019.
55. GANG C, HONGZHE Y, WEI D, et al. Learning to navigate from simulation via spatial and semantic information synthesis with noise model embedding [J/OL] 2019, <https://doi.org/10.48550/arXiv.1910.05758>.
56. SZEGEDY C, LIU W, JIA Y Q, et al. Going deeper with convolutions; proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, F Jun 07–12, 2015 [C]. Ieee: NEW YORK, 2015.
57. SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale imagerecognition[J/OL]2014, <https://doi.org/10.48550/arXiv.1409.1556>.
58. KAIMING H, XIANGYU Z, SHAOQING R, et al. Deep residual learning for image recognition; proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, F Jun 27–30, 2016 [C]. Ieee: NEW YORK, 2016.
59. HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks; proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, F Jul 21–26, 2017 [C]. Ieee: NEW YORK, 2017.
60. ZHAO H S, SHI J P, QI X J, et al. Pyramid scene parsing network; proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, F Jul 21–26, 2017 [C]. Ieee: NEW YORK, 2017.
61. HU S X, CHEN C P, ZHANG A W, et al. A small and lightweight autonomous laser mapping system without gps [J]. J Field Robot, 2013, 30(5): 784–802.
62. WULF O, ARRAS K O, CHRISTENSEN H I, et al. 2D mapping of cluttered indoor environments by means of 3d perception; proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, F Apr 26–May 01, 2004 [C]. Ieee: NEW YORK, 2004.

Chapter 5

Robot Grasping and Interaction



5.1 Overview

Intelligent service robots are identified as a frontier technology to be developed over the next 15 years in the “National Medium and Long-term Science and Technology Development Plan (2006–2020).” In 2012, China formulated the “Twelfth Five-Year Plan for the Development of Service Robot Technology” to support the growth of the robot industry. The application scenarios for robots in real life are becoming increasingly diverse, with autonomous grasping being an essential function for robots to interact with the real world. Whether for industrial robots on assembly lines or service robots in stores and homes, autonomous grasping is crucial for object interaction.

Autonomous robot grasping consists of two steps: the first is grasping pose detection, where the algorithm analyzes specific objects to generate a reliable grasping pose; the second is the motion planning of the robotic arm, which involves planning the arm’s movement to reach the specified grasping pose and complete the task. Grasping pose detection is foundational for achieving this task, relying heavily on computer vision algorithms. The input data for the grasping pose detection algorithm includes images from the robot’s camera, such as RGB and depth images. While RGB images provide rich color information, they lack 3D spatial data. However, as technology advances, the cost of depth cameras has decreased, and many robots are now equipped with RGB-D cameras. The depth information from depth images compensates for the spatial information gap in RGB images, enabling the algorithm to utilize richer data for detecting object grasping poses.

In recent years, algorithms for detecting robot grasping postures have made significant strides. Initially, researchers modeled objects and analyzed their geometric information and features to calculate grasping postures. However, this approach, involving kinematics and dynamics, proved complex to implement. Consequently, the grasping postures generated in models often failed to produce the desired results when applied to real-world objects. With advancements in deep learning and

computer vision, numerous robot grasping posture detection algorithms based on deep learning have emerged. These algorithms learn from manually annotated data, achieving satisfactory results when transferred to real-world applications. Nonetheless, the performance of deep learning-based detection algorithms relies heavily on the availability of rich annotated datasets; without them, the network model struggles to learn accurate parameters.

This chapter first explores robot grasping methods based on analytical and empirical approaches, employing object interaction dynamics to predict grasping postures. Additionally, for deep learning methods, this chapter presents four practical approaches.

If you wish to access the reference code and dataset related to these methods, please scan the SN-branded ShortURLs.

► sn.pub/xhgzn

5.2 Related Work

5.2.1 *Detection of Grasping Methods Based on Analytical Methods and Empirical Methods*

The detection of object grasping methods based on analytical methods mainly uses mathematical and physical geometric models of objects, combined with dynamics and kinematics to calculate the current stable grasping method of the object. Because the interaction behavior between the mechanical gripper and the object is difficult to model, this detection method does not yield good results in real-world applications. The detection of object grasping methods based on empirical methods focuses on using object models and past grasping experience. In these methods, some work involves establishing a database of object models and associating known objects with effective grasping methods. When presented with a current object, these methods search for similar objects in the database to obtain effective grasping methods. This approach shows relatively good application effects in actual environments compared to the analytical method but still lacks generalization ability for unknown objects. Methods [1, 2] parameterize object grasping, listing four parameters for analyzing a grasp: (1) The grab point on the object that the tool center point should align with; (2) Approach vector, describing the 3D angle at which the robot hand approaches the grab point; (3) The direction of the robot's wrist; and (4) The initial configuration of the robot's fingers

The analytical method provides a guarantee for measuring the above four grasping characteristics. However, it is usually based on simplified contact models, Coulomb friction, and rigid body modeling assumptions. Under these assumptions, grasp analysis is practical, but due to its approximate nature, there are still inconsistencies and ambiguities in the analysis, particularly evident in grasping dynamics analysis.

In this case, Bicchi and Kumar [3] found an accurate and easy-to-handle contact compliance model, which requires analyzing some uncontrollable and statically uncertain grasping methods in terms of internal force. For example, for underactuated mechanical hands or cooperative grasping actions, where the controlled degree of freedom (the number of degrees of freedom is less than the number of contact forces). Prattichizzo et al. [4] modeled such systems by introducing a set of springs at the contact points and joints, demonstrating a method for analyzing their flexibility. Rosales et al. [5] used the same compliance model to synthesize feasible and appropriate grasps. In this case, only static, deterministic grasps are considered, transforming the problem of finding the appropriate hand posture into a constrained optimization problem, and introducing compliance to simultaneously solve the constraints of contact reachability, object constraints, and force controllability.

Like many other grasp synthesis analysis methods, this model only studies the simulation of relatively accurate models of hand kinematics and grasping objects. In practical applications, system errors and random errors are inherent in robot systems; these two types of errors arise from sensor noise and inaccuracies in robot kinematics and dynamics, sensors, or object models. Therefore, in experiments, only the approximate position of the object and the robotic hand can be determined, making it difficult to ascertain the exact position of the fingertip. Bicchi and Kumar [3] identified a lack of synthesis grasping methods robust to positioning errors at that time. One research direction explored the concept of independent contact regions defined by Nguyen [6], a set of regions on the object where each finger can be placed independently at any position without losing the force closure property. Roa and Suarez [7] and Krug et al. [8] provided several examples for calculating these regions. Another study on the robustness of end-effector positioning inaccuracies employed locked-in planning. Rodriguez et al. [9] found that locked-in configurations of three-finger manipulators around planar objects are particularly suitable as waypoints for grasping them. Once the robotic hand is in this configuration, whether the fingers are opened or closed, a smooth grasp can be guaranteed without precise finger positioning. Seo et al. [10] fully utilized the phenomenon that there is always a locked configuration before two fingers grasp an object. The whole-body grasp of planar objects is synthesized by first finding a two-contact-point locked configuration and then using additional contact points to constrain the object.

Another common assumption in analysis methods is that the robot can use the exact geometry and physical model of the object, but this is not always feasible. Furthermore, the surface characteristics, friction coefficient, mass, center of mass, and mass distribution of the object may not be known. Some of these object characteristics can be retrieved through interaction: Zhang and Trinkle [11] proposed a method using particle filters to simultaneously estimate the physical parameters of the object and track it while pushing it, with the dynamic model of the object formulated as a mixed nonlinear complementarity problem. Even when the object is obscured and the state estimate cannot be updated through visual observation, the motion of the object can be accurately predicted over time. Although such methods relax some assumptions, they remain limited to simulations [5, 9] or two-dimensional objects [9–11].

5.2.2 *Grasp Detection Based on Deep Learning*

Deep learning methods play a huge role in visual tasks. For the detection of grasping methods of unknown objects, algorithms based on deep learning have made many advancements. The mainstream representation of grasping methods is a rectangular box similar to object detection, but this rectangular box has a rotation angle parameter. A unique grasping posture can be represented using the center coordinates of the rectangular box, the width of the rectangular box, and the rotation angle of the rectangular box. Most of the grasp detection algorithms to date follow a general detection process: detect candidate grasp positions from image data, use convolutional neural networks to evaluate each candidate grasp position, and finally choose the grasp position with the highest evaluation value as the output. Among them, a representative one is the FastR-CNN (Fast Region current neural network) based object detection model proposed by Chu et al. The object grasping method detection model modified by the network has a large number of network model parameters and relatively low real-time performance. Morrison [12] and others proposed a pixel-level object grasping method detection model based on a fully convolutional neural network, outputting four images of the same size as the original image, namely, the grasping value map, width map, sine map of the rotation angle, and cosine map. This model has fewer parameters and higher real-time performance. The detection of grasping methods based on deep learning has good effects in actual scenarios and has a strong generalization ability for unknown objects.

Some researchers hope to use the richer point cloud of 3D information to detect the grasping posture. Varley et al. [13] use a 3D convolutional neural network to complete the point cloud shape of the current object and then generate the grasping posture from the completed point cloud. Varley et al. trained the network with 3D object data collected from different angles, while Mahler et al. [14] created a synthetic point cloud dataset and proposed a grasping posture quality assessment neural network to evaluate the quality of candidate grasping postures.

Some researchers have noticed the commonality between object detection and robot grasping posture detection, so they try to adapt the network of various object detection algorithms to the problem of robot grasping posture detection [12–14]. However, compared to the problem of object detection, the problem of robot grasping posture detection requires higher speed and real-time performance of the algorithm, and the calculation time of the general sliding window grasping posture detection algorithm is longer. In order to reduce the calculation time of the grasping posture detection algorithm, Wang et al. [15] proposed a method to preprocess the candidate grasping posture and discard some of the grasping postures, thereby reducing the number of candidates grasping postures. Some researchers try to abandon the framework of object detection algorithms and use convolutional neural networks to directly generate an optimal grasping posture, but this

method cannot guarantee that the generated unique grasping posture is an effective grasping posture. Some researchers think out of the box, changing the output format of the grasping posture detection algorithm to a pixel-level heat map. Varley et al. [16] use a heat map output to represent the position where the fingers should be placed on the robot's gripper, but this method cannot directly judge whether a grasping method is effective and requires subsequent grasping planning algorithms to judge its effectiveness. Morrison et al. [12] proposed a completely regressive neural network, which generates different heat maps for various parameters in the grasping posture, including the grasping position, rotation angle, and width, and retrieves the rotation angle and width of the grasping posture from the coordinates of the highest quality position, thereby generating the final grasping posture.

Although the robot grasping posture detection algorithm based on deep learning has achieved good results, there are still problems in balancing real-time performance and accuracy. Deepening the width and depth of the neural network can indeed make the network model perform more accurate detection, but at the same time, the huge increase in the number of parameters also prolongs the calculation time of the neural network's forward propagation. The problem of robot grasping posture detection requires high real-time performance. Simplifying the structure of the neural network can ensure high real-time detection, but it needs to sacrifice a part of the detection accuracy, which will directly affect the final success rate of robot grasping. In addition, deep learning algorithms have a great demand for labeled data. If there is not enough labeled data, the neural network of deep learning algorithms cannot learn accurate parameters. However, labeling a rich dataset requires a lot of resources, which poses a great obstacle to the application of robot grasping posture detection algorithms in real life.

5.2.3 Summary

This section introduces several research directions on the current state of domestic and international research on grasping methods, finding that traditional methods of analysis and empirical methods both have significant limitations. Only the grasping method detection based on deep learning performs well in actual scenarios, and it has a strong generalization ability for unknown objects. However, even though the grasping method detection based on deep learning has made remarkable progress, this method is still limited by the large demand for data by deep learning, mainly in two aspects: First, if training is conducted in the traditional way and there is not enough annotated data, the network model cannot achieve satisfactory accuracy; second, when an existing model is transferred to the detection problem of unfamiliar objects, if data collection and annotation for unfamiliar objects are to be carried out, it will consume a lot of manpower.

5.3 Technical Practice

5.3.1 Case Study: Grasping Pose Detection Based on Object Interaction Dynamics

5.3.1.1 Task Description

In robot kinematics, implementing robot grasping pose detection and path planning under uncertain conditions is a difficult task. The accuracy of sensor perception of robots operating in the real world directly affects the accuracy of robot grasping pose detection. To solve this problem, a model is needed to process the original sensor data.



Chapter Knowledge Points

Grasp configuration detecting: Given the object and basic environmental constraints, the goal of grasp configuration detecting is to find the claw pose configuration with the highest grasp quality. For the problem of grasp configuration detecting, existing methods usually belong to one of the following two categories: model-based methods and model-free methods.

Force closure (FC): Counteract any external force and external torque acting on the object using the contact force applied by the fingers (allowing static friction constraints, including frictionless) at the given contact points. In addition, under the above assumptions about fingertip grasping, force closure not only ensures that it can counteract any external force but also allows the object to produce any acceleration.

Motion planning: It is to find a path that meets the constraints between a given position A and position B for the robot, plan the shortest distance for the mobile robot to reach the designated place, or plan a collision-free motion trajectory for the mechanical arm, so as to achieve object grasping, etc.

Convolutional neural networks (CNN): It is a type of feedforward neural network that includes convolutional computation and has a deep structure. It is one of the representative algorithms of deep learning. Convolutional neural networks have the ability to learn representations, can perform translation invariant classification of input information according to their hierarchical structure, perform supervised learning and unsupervised learning, and the sharing of convolution kernel parameters within the hidden layer and the sparsity of inter-layer connections make convolutional neural networks able to learn grid-like features, such as pixels and audio, with a small amount of computation, have stable effects and do not require additional feature engineering for data.

5.3.1.2 Related Work

Grasp Configuration Detecting

Model-based methods [17, 18] usually rely on pre-built general 3D object model grasp databases, these models are provided by GraspIt! [19] Auxiliary tools provide feasible grasping and quality metric set tags. During execution, it is necessary to associate sensor input data with object entries in the database for grasping planning, mainly based on visual and geometric similarity [20–22]. However, under the influence of inaccurate sensor data and a fixed-size database, model-based methods may have poor generalization performance for new objects and objects presented in dense clutter. Compared with model-based methods, model-free methods usually consist of two independent parts, namely, grasp candidate generation and grasp quality metrics.

In the process of grasp candidate generation, the geometric information captured by the sensor will be used as heuristic constraints [23] to build an adaptive grasp configuration sampler on a given object. These candidate grasps will be evaluated through quality metrics. In some modern model-free methods, a large grasp dataset is also needed to train quality metrics based on deep neural networks [14, 24].

3D Computer Vision in Robotic Grasping

In robotic grasping, the uncertainty of perception has a significant impact. The robot's gripper needs to interact with objects in 3D space, so precise and detailed 3D visual analysis is the key to successful grasping. Driven by the success of deep neural networks in various 3D computer vision tasks [25–28], there are now some experiments that combine 3D computer vision technology with grasp planning [13, 14, 24, 29, 30]. Zeng et al. [18] introduced a channel that can detect grasp poses in dense clutter. They use multi-view depth input to offset the shortcomings of depth perception, but their method is particularly dependent on the CAD model of the grasped object, which means that the method will not be able to generalize to new objects. Varley et al. [13] proposed convolving voxelized 3D meshes from point clouds (3D-CNN) to obtain the geometric representation of the grasped object. This representation is then input into the grasp generation model. Using 3D-CNN to improve the geometric analysis of grasp poses. However, the main disadvantage of this method is that the runtime and memory complexity grow cubically with the resolution of the input 3D voxel [36]. Therefore, the input must be limited to a relatively low resolution. In addition, the sparsity of point clouds may disrupt the neural network's learning of meaningful features of grasp geometry, as most voxels will not be occupied by any points. Ten et al. [29] proposed a grasp pose detection algorithm (GPD) that designed several projection features on normalized point clouds to build a CNN-based grasp quality assessment model and achieved excellent performance in grasping objects from dense clutter. However, due to the network architecture and handmade depth features, when the input point cloud is overall sparse, the

GPD algorithm will have serious overfitting and performance degradation problems. Moreover, in most real-world grasping situations, it may be difficult to obtain a relatively comprehensive point cloud, especially when the clutter is highly occluded. Using 2D or 2.5D input is not enough for geometric analysis. By introducing the Point Cloud Network (PointNet) [31] for 3D representation learning and fine-grasp quality labels for supervision, the PointNet-based Grasp Pose Detection algorithm (PointNetGPD) outperforms these results in terms of grasp performance and efficiency.

5.3.1.3 Technical Methods

Algorithm Flow

The PointNetGPD algorithm flow is shown in Fig. 5.1. Given the original RGB-D data input by the sensor, the depth map is converted into a point cloud; then, according to the necessary geometric constraints, some candidate grasp poses are sampled. For each candidate grasp, the point cloud inside the gripper is cropped and transformed into the local coordinate system of the gripper. Finally, the candidate grasps are input into the grasp quality evaluation network to obtain their respective scores, and the candidate grasp with the highest score is adopted and executed.

Grasp Evaluation Problem Description

Given an object o , with a friction coefficient r between it and the gripper, the geometric center of the object is M_0 , the 6D pose is W_0 , and the grasping state is set as $s = (M_0, W_0, r)$; a single grasp $g = (p, r)$, where $p = (x, y, z)$, $r = (r_x, r_y, r_z)$ represent the grasping position and orientation of the gripper, respectively. All spatial attributes are in the camera coordinate system; let the grasp quality be $Q(s, g)$, a grasp metric

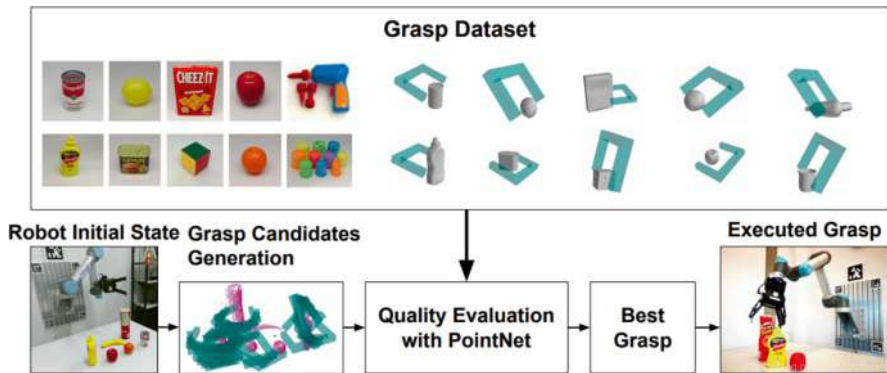


Fig. 5.1 The PointNetGPD algorithm flow

$Q_d(P, g)$ is learned from the point cloud data through the given grasp g and sensor observation data P , and the category is assigned through $Q(s, g)$.

To train such a grasp metric and ensure its effectiveness and generalization performance, large-scale data sampling is required. This technical case chooses the PointNet network to generate point cloud and grasp pose datasets, and trains the evaluation model on the dataset.

Grasp Data Generation

1. Sampling: By randomly finding two surface points p_1 and p_2 of the object in the point cloud data as the contact points of the gripper, a grasp $g((p_1 + p_2)/2, R)$ is constructed by randomly sampling a grasp angle of $0-90^\circ$; then by judging whether the gripper's grasping action will cause unnecessary collisions, unsuitable grasps are removed; finally, the remaining grasp poses are transformed from the grid coordinate system to the point cloud coordinate system.
2. Scoring: The grasp score is evaluated according to the Force Closure method (FC) and the Grasp Wrench Space analysis method (GWS). Since the FC method requires the use of the friction coefficient r , this technical case determines the minimum friction coefficient $r_{\min}$ of the two points taken by a grasp by continuously magnifying the friction coefficient and takes $1/r_{\min}$ as the score of the grasp. The friction coefficient grasp metric uses GWS, and finally the results of the grasp quality $Q_{fc}(s, g)$ under the FC metric and the grasp quality $Q_{gws}(s, g)$ under the GWS metric are weighted to get the final result. Since Q_{gws} is much larger than Q_{fc} , this technical case sets the coefficients (α, β) to $(1.0, 0.01)$.

$$Q(s, g) = \alpha Q_{fc}(s, g) + \beta Q_{gws}(s, g) \quad (5.1)$$

Scoring examples are shown in Fig. 5.2, where (a) has a friction coefficient value of 0.4, and (b) has a friction coefficient value of 2.0. It can be observed that

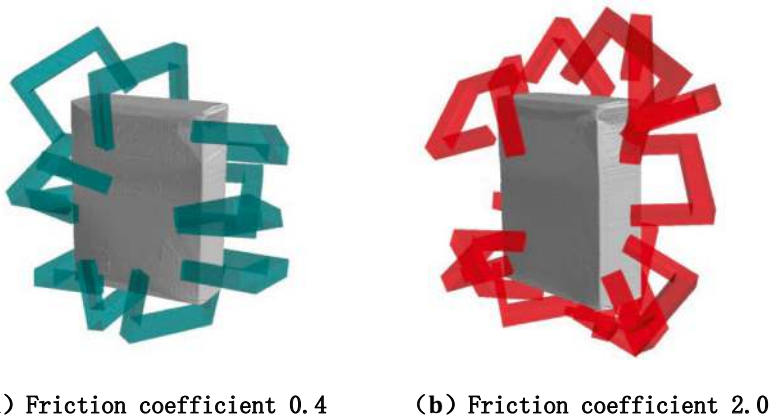


Fig. 5.2 Examples of dataset labels Q_{fc}

on simple geometric objects, different friction coefficients make a big difference in grasp robustness.

Network Framework and Training

The input of the grasp quality evaluation network is the point cloud information in the gripper’s internal space, not the entire point cloud, which significantly improves the learning and inference efficiency of the network. The network structure is shown in Fig. 5.3. Compared with other CNN-based grasp quality evaluation networks, this network has only approximately 1.6 million parameters, making it a lightweight network.

First, the point cloud is converted into the unified gripper coordinate system shown in Fig. 5.4, eliminating the ambiguity caused by different settings in parameters in different experiments (especially cameras). Specifically, the gripper’s gripping direction, closing direction, and orthogonal direction are regarded as the x , y , and z axes, respectively, and the origin is at the center of the bottom of the gripper. In this coordinate system, the coordinates of N points in the gripper’s internal space are input into the network to estimate the level of grasp quality.

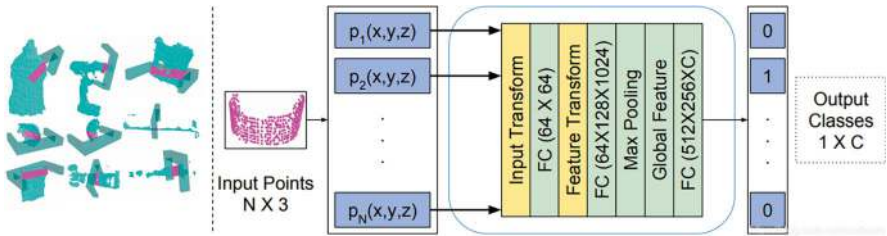


Fig. 5.3 Structure of the grasp quality evaluation network based on PointNet

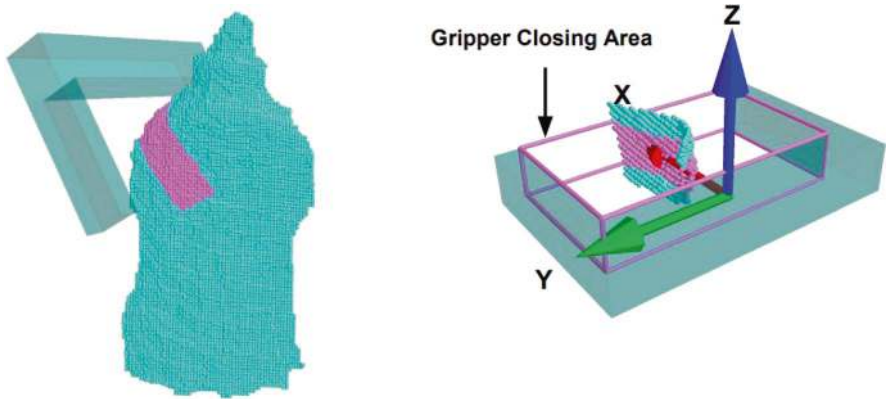


Fig. 5.4 Representation of point cloud information for grasping in the gripper coordinate system

Given a grasping pose and the original point cloud, the point cloud information in the gripper's closing space in the original point cloud represents a grasp. The point cloud information is transformed to the coordinate system where the gripper is located. Through multiple spatial transformations and feature extractions, the final global features are obtained for dividing the quality of the grasp.

In terms of network training, the multiple grasping information obtained from the "grasping data generation" sampling is used as the training set for network input. The quality assessment of each grasp is a quantitative result, not a binary value, so it is suitable for assigning the quality of various types of grasps. To balance the quality of different grasps, different friction coefficients are invoked, and an equal number of grasps are sampled, setting Q_{fc} to $\left\{ \frac{1}{0.4}, \frac{1}{0.45}, \frac{1}{0.5}, \frac{1}{0.8}, \frac{1}{1.2}, \frac{1}{1.6}, \frac{1}{2.0} \right\}$.

Choosing to generate point clouds based on 47 types of YCB object models instead of point clouds rendered from CAD models ensures the network's generalization ability for grasping tasks in the real world.

In terms of training details, the network uses C-class cross loss as the target of the classifier, and the entire network structure is optimized using the Adam optimizer. The initial values of all network parameters are sampled from a 0-mean Gaussian distribution. While ensuring that all points are within the gripper's graspable area, random offsets are added to the point cloud to increase the total amount of data.

Generation of Grasping Candidates

The generation of grasping candidates is a prerequisite for grasping planning. This technical case uses the GPG algorithm to perform heuristic grasping sampling on point cloud data. In addition, to reduce the collision between the generated grasps and the support surface, some modifications have been made to the original GPG algorithm: first, exclude candidate grasps that are too close to the support surface; then remove candidate grasps that are far from the support surface during the grasping process. For grasps with collisions, try to move the grasping point in the opposite direction until the collision disappears. If there are still some points in the gripper's graspable area after applying the GPG algorithm, the adjusted grasps are marked as non-collision grasps.

5.3.1.4 Results Display

Simulation Experiment Results

This technical case chooses the GPD algorithm as the benchmark algorithm for comparison. In addition, to observe the stability on sparse point clouds, this technical case provides two types of point cloud input data; single-view point clouds

obtained from the camera in front of the object, and complete point clouds registered from point clouds under all views. After the point cloud data is prepared, remove those candidate grasps where the number of points between the grippers is < 50 , and up/down sample the remaining candidate point clouds to make the number of points 1000. Carry out 3-class recognition (3-class) and 2-class recognition (2-class) experiments, respectively. For 2-class recognition, scores $> 1/0.6$ are considered positive; For 3-class recognition, the threshold for positive scores is set to $1/0.5$ and $1/1.2$.

The test accuracy obtained during training is shown in Fig. 5.5. Among them, the best results at 200 epochs (training rounds) are compared in Table 5.1. These bold values indicate the best results. First, the PointNetGPD algorithm is better than all GPD algorithms, even on the most difficult single-view point clouds, the PointNetGPD algorithm still has an average improvement of 4.79% compared to the best GPD algorithm. In addition, as can be seen from Fig. 5.5, the GPD algorithm is prone to overfitting on the dataset, and even with the use of the random dropout (DropOut) strategy, there is still a gap between this algorithm and the PointNetGPD algorithm. Part of the reason here is the issue of the number of parameters. The PointNetGPD algorithm has fewer parameters and performs better, which indicates that the network proposed in this technical case is more effective for geometric analysis of sparse point clouds. Furthermore, in the 3-class recognition experiment, the accuracy of the category with the highest grasping quality is higher than the category accuracy in the 2-class recognition experiment, which means that grasps with higher grasping scores are easier to recognize.

Robot Grasping Experiment in Actual Environment

This technical case evaluated the reliability and effectiveness of the PointNetGPD algorithm in two real grasping environments of robotic arms: objects placed separately and objects stacked. The experiment used the UR5 robotic arm and the Robotiq three-finger gripper. As shown in Fig. 5.6a, when the gripper grips the object, there are only two contact surfaces. The Kinect2 depth sensor was used in the experiment, with only single-view point cloud data.

This technical case selected 22 objects from the YCB dataset, of which 11 exist in the grasping dataset and the remaining 11 are new. Sixteen objects were selected from the 22 objects to build two datasets for stacked object removal, as shown in Fig. 5.6b. This technical case provides the results of the 2-class PointNetGPD algorithm and the 3-class PointNetGPD algorithm and compares them with the 15-channel GPD algorithm. The entire system was developed under the ROS framework, using a fast hybrid progressive inverse dynamics operator BioIK built into MoveIt!.

1. Objects placed separately: Each object was tested 10 times, with the orientation of the object random each time. If the gripper cannot successfully grasp, or if no collision-free grasping pose can be found within 5 min, the grasp is considered

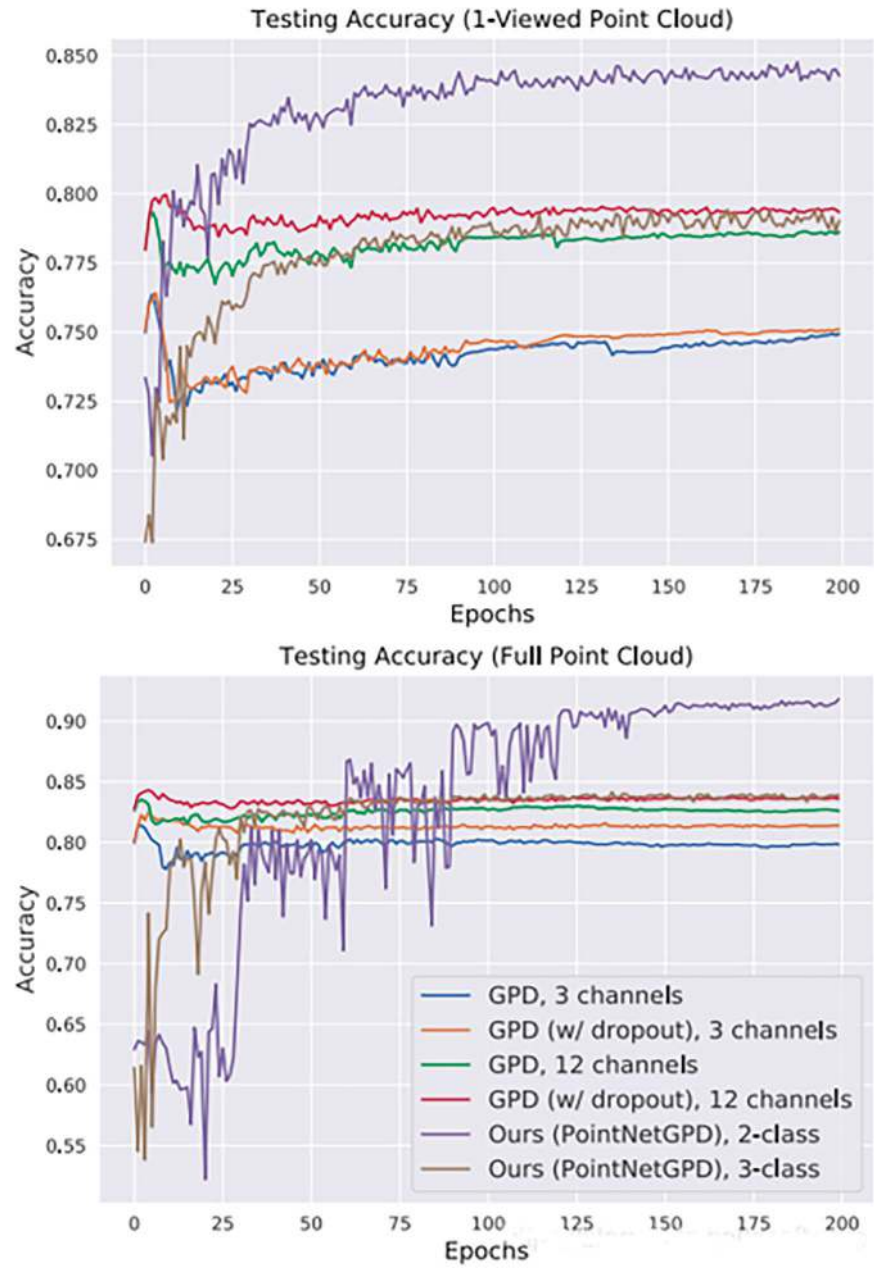


Fig. 5.5 Recognition accuracy under different models and configurations (see color figure in front of the text). (a) Test accuracy (single-view point cloud input); (b) Test accuracy (full-view point cloud input)

Table 5.2 Analysis of single object grasping practice results

Method	Avg	Cleanser bottle	Mug	Meatcan	Tomato soup can	Banana	Toy power drill	Chain	Mustard bottle	Wood block	Screwdriver
GPD	49%	100%	30%	60%	90%	20%	80%	0%	90%	90%	20%
Pointnet GPD 2-class	81%	100%	50%	80%	100%	90%	70%	60%	100%	90%	70%
Pointnet GPD 3-class	82%	90%	70%	70%	100%	90%	80%	60%	90%	90%	80%

Table 5.3 Analysis of stacked object grasping practice results

	GPD		Pointnet GPD 2-class		Pointnet GPD 3-class (best class)		Pointnet GPD 3-class (second class)	
	Success rate	Completion rate	Success rate	Completion rate	Success rate	Completion rate	Success rate	Completion rate
Set 1	84.83%	95.00%	86.54%	94.08%	89.33%	100.00%	52.10%	100.00%
Set 2	61.13%	81.50%	61.07%	84.38%	66.20%	95.00%	43.75%	37.50%

5.3.1.5 Summary

1. This technical case introduces the PointNetGPD algorithm that can directly process point cloud data. This algorithm is trained on a dataset containing 350,000 real object point cloud grasping data based on the YCB model library, and it improves the grasping effect compared to the current mainstream GPD algorithm.
2. The integration of candidate grasping pose generation and grasping quality assessment network is conducive to the implementation of end-to-end grasping pose detection. The introduction of scene segmentation in the case of multiple object stacking can better separate multiple objects geometrically.

5.3.2 Case Study: Robot Grasping Based on Active Learning

5.3.2.1 Task Description

Robot grasping is a basic function of intelligent robots and a challenging task. Many grasping pose detection algorithms have been proposed by researchers today. However, for these deep learning algorithms, a rich dataset is essential. Robots may move in different environments. When the environment changes, it is necessary to create a new dataset and retrain the model to maintain the performance of the network model. However, dataset annotation is a very resource-consuming process. Active learning aims to alleviate the dependence of deep learning algorithms on a large amount of annotated data. The main approach is to select the most informative data from the unlabeled dataset for annotation, without the need to annotate the entire dataset. The work of this technical case mainly includes the following aspects.

Discriminative Active Learning Strategy

A discriminative active learning strategy for robot grasping is proposed. This strategy uses a shared encoder to obtain the latent features of labeled and unlabeled data and establishes a discriminator to estimate the similarity of each unlabeled data with the labeled dataset. The data with the lowest similarity is considered the most informative data.

Deep Separable Convolution Grasping Pose Detection Network

The current robot grasping pose detection algorithm has a balance problem between accuracy and real-time performance. This technical case uses deep separable convolution to improve the grasping pose detection network, limiting the growth of parameter quantity while achieving higher accuracy, thus ensuring real-time

detection. Combined with the discriminative active learning strategy, this technical case designs a complete robot grasping training application framework.



Chapter Knowledge Points

Unstructured Environment: The surface (ground, wall, obstacle surface) material properties (surface material, roughness, stiffness, strength, color, reflection, temperature, etc.) are uneven, the structure and size changes are irregular and unstable, and the environment information (obstacles, lighting, sound, gas, radiation, wind force, interference, etc.) is non-fixed, unknown, and indescribable working environment.

Target Recognition: It is to use a computer to realize human visual functions. Its research goal is to enable the computer to have the ability to recognize the surrounding environment from one or more images or videos (including the perception, recognition, and understanding of the objective world's 3D environment).

Active Learning: A machine learning framework in which the learning algorithm can interactively query users to label new data points with real labels. The process of active learning is also known as optimal experimental design.

6D Pose: 6D refers to six degrees of freedom, representing three degrees of freedom for displacement (also called translation), and three degrees of freedom for spatial rotation (rotation), together called pose. Pose is a relative concept, referring to the displacement and rotation transformation between two coordinate systems. The 6D pose of an object and the 6D pose of a camera are similar, the difference lies in the transformation from which coordinate system to the camera coordinate system.

5.3.2.2 Related Work

Grasping Pose Detection

The goal of robot grasping pose detection is to output the effective grasping pose of a given object. Although intuitively it only needs to generate a suitable grasping method, it needs to consider many factors, such as the shape of the object, physical properties, and the interaction between the robot's gripper and the object. It should be noted that the robot grippers involved in this technical case are all assumed to be parallel grippers.

In early research, researchers tried to find a method to generate the object's grasping pose from the physical model of the robot's gripper and the object, and the dynamics of the interaction between the gripper and the object. This method is called the analytical-based grasping pose detection algorithm. Although deep learning-based robot grasping pose detection algorithms have achieved good results,

there are still problems in balancing real-time performance and accuracy. Increasing the width and depth of the neural network can indeed make the network model perform more accurate detection, but the huge increase in the number of parameters also prolongs the calculation time of the neural network's forward propagation, and the robot grasping pose detection problem requires high real-time performance. Simplifying the structure of the neural network can ensure high real-time detection, but it requires sacrificing some detection accuracy, which will directly affect the final success rate of robot grasping. In addition, deep learning algorithms have a great demand for labeled data. If there is not enough labeled data, the neural network of the deep learning algorithm cannot learn accurate parameters. However, labeling a rich dataset requires a lot of resources, which poses a great obstacle to the application of robot grasping pose detection algorithms in real life.

Active Learning

Active learning aims to reduce the workload of data labeling, mainly by selecting a part of the most informative data from the dataset, only labeling this part of the data for training the neural network, thereby saving the resources consumed by labeling the remaining data. This process is different from weakly supervised learning. Active learning expects to find the redundant part of the dataset and discard them, but the labels used on the retained data are still from strongly supervised learning. How to define the information content of a certain data is the key to the active learning strategy. The most intuitive and natural active learning strategy is the random selection strategy. This strategy randomly selects a part of the data from the unlabeled data for training, without any selection basis. This strategy is generally used to measure the performance of other active learning strategies.

In addition to the information content inherent in the data, the neural network of deep learning also contains various parameters that can represent the information content of a certain data. This strategy uses the posterior probability of the classification neural network to calculate a parameter that represents the uncertainty of the network model for the input data, and this uncertainty is regarded as the information content of the data. If a neural network is more uncertain about a piece of data, then this piece of data should be selected to be included in the next round of training.

Although there are currently many active learning strategies, these strategies face two main problems when applied to the issue of robot grasping pose detection: first is the problem of incompatibility. Most of the current active learning strategies are designed for classification problems, while the results needed to be output by the robot grasping pose detection algorithm involve image coordinate positions, causing the grasping pose detection network to involve regression problems to varying degrees. This makes it difficult, if not impossible, to apply active learning strategies specifically designed for classification problems directly to robot grasping pose detection algorithms. Second is the issue of parameter quantity. For general computer vision problems, such as object detection and semantic segmentation, the parameter quantity of the algorithm network is relatively large. Therefore,

researchers do not consider the issue of parameter quantity too much when designing active learning strategies, and many active learning strategies applied to these problems contain a large number of parameters. However, the problem of robot grasping pose detection has a high requirement for real-time performance, so the parameter quantity of the grasping pose detection network is often limited. If an active learning strategy with a large number of parameters is used, it will cause additional resource consumption. Therefore, active learning strategies for robot grasping need to consider multiple issues.

5.3.2.3 Technical Methods

The purpose of robot grasping detection is to detect the grasping pose of a given object. Even though there are many excellent deep learning-based robot grasping pose detection algorithms to date, the performance of the grasping pose detection model will inevitably decline when facing new environments and objects, and in severe cases, it may lead to incorrect grasping pose estimation results. To obtain more accurate grasping pose detection results, a reliable and common practice is to collect and label data for new objects, and then retrain the model with the labeled data. However, when the environment and objects change frequently, executing the process of data collection, data labeling, and training each time will consume a lot of human resources. Especially the data labeling process requires experienced labelers to work for a long time, and the accuracy of data labeling will directly affect the final effect of the grasping pose detection model. Therefore, the data labeling process is a link that should be optimized.

The idea of active learning is to select the most informative data from unlabeled data and only label this part of the data for deep learning network training, rather than labeling the entire dataset. There may be redundancy in a large amount of unlabeled data, and redundant data contributes little to the performance improvement of the neural network. If appropriate data is selected, the performance of the network model trained with part of the labeled data can also be comparable to the network model trained with the entire dataset. However, when applying the idea of active learning to the problem of robot grasping, there are mainly two problems. First, many current active learning strategies cannot be applied to the robot grasping pose detection network. The grasping pose detection network involves regression algorithms, and active learning strategies specifically designed for classification tasks generally need to use the probability vector output by the classification network, so these active learning strategies cannot be applied to the grasping pose detection network. When designing an active learning strategy for robot grasping, the strategy must be applicable to regression networks. Secondly, active learning strategies need to have a smaller parameter quantity. For the real-time nature of grasping pose detection, the parameter quantity of the robot grasping pose detection network is generally small. If an active learning strategy consists of a large number of parameters, then applying it to the grasping pose detection network will consume a lot of additional computing resources. Therefore, the active learning strategy for the robot

grasping pose detection network needs to have certain restrictions on the parameter quantity.

Based on the above two issues, the existing active learning strategies face difficulties when migrating to the problem of robot grasping pose detection. This technical case proposes a discriminative active learning strategy, which is independent of network output and can be applied to both classification networks and regression networks. Therefore, the strategy proposed in this technical case can be adapted to the robot grasping pose detection network. In addition, this strategy has fewer network parameters, which are negligible even for robot grasping pose detection networks with very small parameter quantities. Therefore, the active learning strategy proposed in this technical case will not consume too much additional computational resources.

Network Structure Design

Figure 5.7 shows the overall network structure of the discriminative active learning strategy proposed in this technical case, which is divided into two parts: the robot grasping pose detection network and the discriminative active learning data selection strategy network. The input of the robot grasping pose detection network is RGB images and depth images, and the output is the result of grasping pose detection, including the coordinates of the center point of the grasping rectangle, the width of the grasping rectangle, and the rotation angle. The robot grasping pose detection network needs labeled data for training. Correspondingly, the discriminative active learning data selection strategy network accepts RGB images and depth images as input, and the output is the judgment result of the active learning selection strategy network on the size of the input data information. In principle, the active learning strategy network proposed in this technical case can be applied to any robot grasping pose detection network and does not limit the structure of the robot grasping pose detection network. In the design stage of this technical case and the

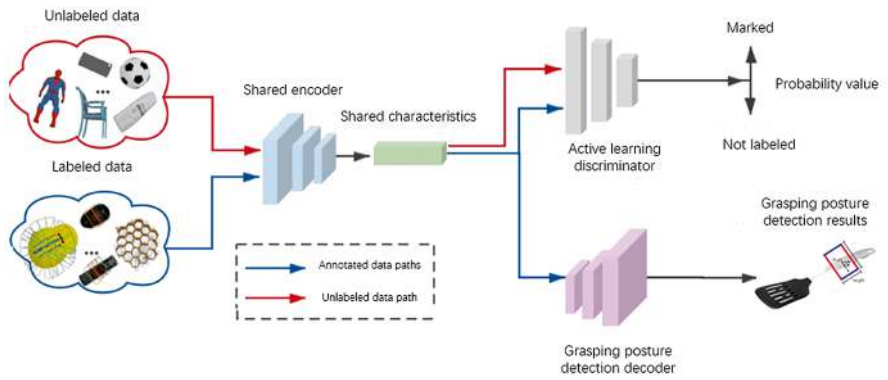


Fig. 5.7 Discriminative active learning network structure (see color picture before the text)

subsequent experimental process, the design of the robot grasping pose detection network refers to the Generative Grasping CNN (GG-CNN) network structure proposed by Morrison et al. [12], mainly for three reasons.

1. The real-time performance of GG-CNN is very high. This grasping pose detection network abandons the framework of traditional object detection models, uses a fully convolutional neural network, and treats the robot grasping pose detection problem as a complete regression problem. Therefore, this grasping pose detection network has very few parameters, and even if a computer with not very high configuration is used to run this grasping pose detection network, a fast detection result can still be obtained. For a robot grasping task in a real scene, GG-CNN can meet the requirements of real-time detection.
2. The detection accuracy of GG-CNN is acceptable. Although this grasping pose detection network uses fewer parameters to greatly improve the speed of grasping pose network detection, its detection accuracy can still reach a high level.
3. GG-CNN has a streamlined network structure. The overall structure of this grasping pose detection network only includes the stacking of convolutional layers and deconvolutional layers, which can detect the grasping pose of the objects contained in the input image. Thanks to the streamlined structure of GG-CNN, this technical case can focus more on the design of active learning strategies, and the implementation of active learning strategies and comparisons with other strategies are also more effective.

The above text categorizes the network structure of the discriminative active learning strategy proposed in this technical case. If the discriminative active strategy network is divided in more detail structurally, the network structure proposed in this technical case is mainly divided into three parts: shared encoder, active learning discriminator, and grasping pose detection decoder. This technical case will introduce these three parts in more detail.

Shared Feature Encoder

For the encoding of input data, this technical case adopts the form of a shared encoder, allowing the robot's grasp posture detection network and the active learning selection strategy network to share the same encoder. The shared encoder, as the entrance part of the network structure, mainly accepts input images and extracts latent features from them. The latent features of the input data are also shared by the subsequent active learning discriminator and the grasp posture detection decoder. The advantages of this design are mainly two-fold:

First, as mentioned above, for the real-time effect of grasp posture detection, the parameter volume of a general robot's grasp posture detection network tends to be small. By using a shared feature encoder, the amount of additional parameters needed for the active learning selection strategy network can be significantly reduced. Moreover, by utilizing the data features used in the robot's grasp posture detection network, the active learning selection strategy network can better grasp

the amount of information the input data has for the grasp posture detection network, thus making more accurate data selections.

Second, using a shared feature encoder can make full use of all the data in the dataset, including labeled and unlabeled data. If separate feature encoders are used, the feature encoder in the active learning strategy network can use all labeled and unlabeled data for training, but the feature encoder in the robot's grasp posture detection network can only use labeled data for training. Training the shared feature encoder with all labeled and unlabeled data, even if the number of data in the labeled data pool is small, the shared feature encoder can still achieve sufficient training effect. This point has also been confirmed in the subsequent experimental part of this technical case.

For the network design of the robot's grasp posture detection network GG-CNN used in this technical case, the feature encoder structure is composed of convolutional layers with different receptive fields. The specific structure and input/output of the shared feature encoder used in the network design and experiments of this technical case are shown in Table 5.4. The input data size is $4 \times 300 \times 300$, where 4 represents 4-channel input, which is connected by 3-channel input of RGB image and 1-channel input of depth image, and the size of the input image is 300 pixels $\times$ 300 pixels. After the first convolutional layer, the size of the input image becomes 100 pixels $\times$ 100 pixels. After processing by the Rectified Linear Unit (ReLU), the first feature map of the input data is obtained, with specific parameters of 32 channels 100 pixels $\times$ 100 pixels. Similarly, after the combination of two convolutional layers and a linear rectification layer, the feature map obtained from the input image is 25 pixels $\times$ 25 pixels, with 8 channels. The outputs of the active learning discriminator and the grasp posture detection decoder are different. The active learning discriminator deals with classification problems, while the grasp posture decoder deals with regression problems. Therefore, the active learning discriminator and the grasp posture detection decoder have their own different ways of processing and using the features obtained from the shared feature encoder. The active learning discriminator cannot directly use the feature map obtained from the convolutional layer of the shared encoder, so it needs to preprocess the feature map to obtain the feature vector; while the grasp posture detection decoder is composed of deconvolutional layers, so it can directly use the feature map results obtained from the convolutional layer of the shared encoder. This technical case will next explain the functions and detailed structures of both.

Table 5.4 Active learning shared feature encoder network parameters

Name	Input	Output
Conv1	$4 \times 300 \times 300$	$32 \times 100 \times 100$
ReLU	$32 \times 100 \times 100$	$32 \times 100 \times 100$
Conv2	$32 \times 100 \times 100$	$16 \times 50 \times 50$
ReLU	$16 \times 50 \times 50$	$16 \times 50 \times 50$
Conv3	$16 \times 50 \times 50$	$8 \times 25 \times 25$
ReLU	$8 \times 25 \times 25$	$8 \times 25 \times 25$

The active learning discriminator is the core part of the discriminative active learning strategy proposed in this technical case. Its output is the basis for the discriminative active learning strategy’s judgment on the amount of information contained in the input data. After the input data is encoded by the shared feature encoder, a feature vector representing the data is obtained. The active learning discriminator takes this feature vector as input and finally outputs a value between 0 and 1. This value represents the similarity of the current input data to the data in the labeled data pool. The closer this value is to 1, the higher the similarity of the input data to the data in the labeled data pool, which means that there is similar data in the labeled data pool, and this data will not help much in improving the performance of the grasp posture detection network in subsequent training. On the contrary, the closer the value output by the active learning discriminator is to 0, the lower the similarity of the input data to the data in the labeled data pool, and the more abundant the information contained in the input data, which will have a greater promotion effect on the performance improvement of the subsequent grasp posture detection network.

The structure of the active learning discriminator designed in this technical case adopts a multi-layer perceptron structure. More specifically, the discriminator consists of three fully connected layers, and a rectified linear unit (ReLU) is added between each layer. For the robot grasp posture detection network GG-CNN used in this technical case, the active learning discriminator shares the feature extraction layer of this grasp posture detection network. In order to make full use of the feature extraction information of different receptive fields, this technical case combines the different feature maps obtained from the three feature extraction layers of GG-CNN, performs pooling operations on them, and merges them into a feature vector, which is then input into the active learning discriminator. The above feature fusion operation is shown in Fig. 5.8. The feature maps obtained from three different convolutional layers are first processed by the global average pooling layer (GAP), converted into 1×1 feature vectors, and then calculated by the fully connected layer to obtain three feature vectors of equal length. Finally, these three feature vectors are

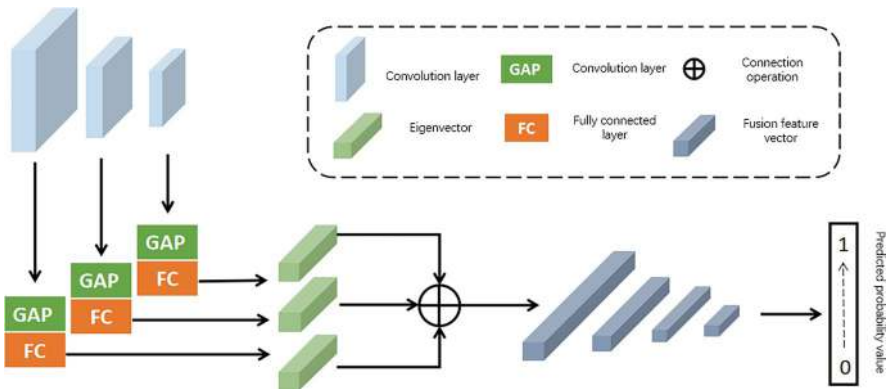


Fig. 5.8 Discriminative active learning strategy feature fusion process (see color figure before the text)

connected and merged into a feature vector, which is input into the active learning discriminator.

The specific feature sizes of the feature fusion process of different feature maps are shown in Table 5.5. The 32×100×100, 16×50×50, and 8×25×25 feature maps of different channel numbers and sizes obtained from the shared feature extraction layer are processed by the 100×100, 50×50, and 25×25 global average pooling layers, respectively, and transformed into 32×1×1, 16×1×1, and 8×1×1 sizes, that is, 32-dimensional, 16-dimensional, and 8-dimensional feature vectors. The feature vectors of different dimensions are processed by the fully connected layer and the rectified linear unit (ReLU) to unify into 16-dimensional feature vectors. Finally, the three 16-dimensional feature vectors are connected and merged into a 48-dimensional feature vector.

This technical case processes the shared features in the network structure according to the above process and then provides the processed features to the active learning discriminator for use. The detailed structure of the active learning discriminator is shown in Table 5.6. The 48-dimensional feature vector goes through three fully connected layers and two linear rectification layers to finally get a 1×1×1 vector, which is the final prediction probability value of the active learning discriminator. In order to make the range of this output value between 0 and 1, the S function, i.e., the Sigmoid function layer, is used at the end of the network structure to process the predicted value and limit the prediction result to between 0 and 1.

Even if a fully connected layer with a large number of parameters is applied in the feature values obtained by the pre-processing shared feature encoder and the active learning discriminator, because the size of the input feature vector is determined by the shared feature encoder, and the shared feature encoder also serves the robot grasping posture detection network, the overall parameter quantity of the active learning discriminator can also be effectively limited. Compared with the

Table 5.5 Feature map feature fusion parameters

Name	Feature map 1	Feature map 2	Feature map 3
Input	32×100×100	16×50×50	8×25×25
GAP	32×1×1	16×1×1	8×1×1
FC	16×1×1	16×1×1	16×1×1
ReLU	16×1×1	16×1×1	16×1×1
Concatenate	48×1×1		

Table 5.6 Active learning discriminator network parameters

Name	Input	Output
FC1	48×1×1	24×1×1
ReLU	24×1×1	24×1×1
FC2	24×1×1	12×1×1
ReLU	12×1×1	12×1×1
FC3	12×1×1	1×1×1
SIGMOID	1×1×1	1×1×1

grasping posture detection network GG-CNN with fewer parameters, the discriminator required in the discriminative active learning strategy proposed in this technical case has very few parameters and does not consume too many computing resources.

The Grasp Pose Detection Decoder is the output component of the active learning strategy target task proposed in this technical case. This decoder corresponds to the shared feature encoder, deconvolving the shared features obtained from the shared feature encoder, ultimately restoring to the size of the input image, and outputting the parameter values corresponding to the robot's grasp pose. In the design of the robot, grasp pose detection network GG-CNN used in this technical case, the deconvolution layer is first used to enlarge the feature image in scale, and then four different convolution layers are used to process the feature map obtained from the deconvolution layer. The final output of the network is four separate $1 \times 300 \times 300$ size images, divided into grasp value image, grasp width image, grasp angle sine value image, and grasp angle cosine value image, 300×300 corresponds to the size of the input image, that is, 300 pixels $\times$ 300 pixels. According to the highest value point coordinates on the grasp value image, the corresponding grasp width and grasp angle can be obtained on the grasp width image, grasp angle sine value image, and grasp angle cosine value image. More specifically, as shown in Table 5.7, the Grasp Pose Detection Decoder only uses the final $8 \times 25 \times 25$ size feature map obtained from the shared feature encoder, not the three different size feature maps used by the active learning discriminator mentioned above, and it does not need to preprocess the feature map, it can be directly input into the Grasp Pose Detection Decoder. After processing through multiple deconvolution layers and linear rectification layers, the network outputs a $32 \times 300 \times 300$ feature map, and four different convolution layers use the same feature map to output the corresponding four result images of the robot grasp pose detection. Overall, the shared feature encoder and the Grasp Pose Detection Decoder form a fully convolutional neural network.

1. Discriminative Active Learning Data Selection Strategy: The previous section has detailed the network structure of the discriminative active learning strategy for robot grasping proposed in this technical case. Next, we will introduce the process of the discriminative active learning strategy selecting the data that

Table 5.7 Grasp Pose Detection Decoder parameters

Name	Input	Output
Deconv1	$8 \times 25 \times 25$	$8 \times 50 \times 50$
ReLU	$8 \times 50 \times 50$	$8 \times 50 \times 50$
Deconv2	$8 \times 50 \times 50$	$16 \times 100 \times 100$
ReLU	$16 \times 100 \times 100$	$16 \times 100 \times 100$
Deconv3	$16 \times 100 \times 100$	$32 \times 300 \times 300$
ReLU	$32 \times 300 \times 300$	$32 \times 300 \times 300$
Conv1	$32 \times 300 \times 300$	$1 \times 300 \times 300$
Conv2	$32 \times 300 \times 300$	$1 \times 300 \times 300$
Conv3	$32 \times 300 \times 300$	$1 \times 300 \times 300$
Conv4	$32 \times 300 \times 300$	$1 \times 300 \times 300$

needs to be labeled from the unlabeled data. For a clearer explanation, the robot grasping posture detection network and the discriminative active learning data selection network will be explained as two separate networks in the following explanation.

2. **Optimizer:** To ensure fairness in subsequent comparative experiments, the Adam optimizer, which is the same as the original robot grasping posture detection network GG-CNN, is used in the selection of the optimizer for the robot grasping posture detection network. The same Adam optimizer was tried for the discriminative active learning network, but it was found that the training effect was not good and the discriminative active learning network could not converge normally. After several attempts, the stochastic gradient descent (SGD) was finally chosen as the optimizer used in the discriminative active learning network.

5.3.2.4 Results Display

The structure, data selection method, dataset used, training details, evaluation criteria, and comparison methods of the discriminative active learning strategy for robot grasping have been introduced above. Next, we will explain and display the results of several comparative experiments, and analyze and evaluate the experimental results.

Results of the Cornell Grasping Dataset

In order to know the performance of the original robot grasping posture detection network GG-CNN, the performance of the original GG-CNN on the Cornell grasping dataset is first tested. As mentioned above, for the fairness of the experiment, the data augmentation strategy adopted in the original GG-CNN paper was not used when training GG-CNN, so the performance test results are different from those in the original paper. It should be emphasized that all the data in the Cornell grasping dataset was used when training the original GG-CNN. In terms of dataset division, 80% of the data in the GG-CNN grasping dataset was used as the training set, and the remaining 20% of the data was used as the test set.

The experimental results are shown in Fig. 5.9, where the x-axis represents the number of epochs in the neural network training process, and the y-axis represents the accuracy of the current network model in predicting grasping postures on the test set. After 200 rounds of training, the accuracy of the original robot grasping posture detection network GG-CNN fluctuates around 80%. Therefore, if the discriminative active learning selection strategy for robot grasping can select a part of the data to make GG-CNN reach a performance of 80% or above, it can prove the effectiveness of the proposed discriminative active learning selection strategy.

In the comparative experiment of active learning strategies, 80% of the Cornell grasping dataset was used as the training set data, and the remaining 20% of the data was used as the test set data. Before all training began, all data in the training set was

considered as unlabeled data. Here, an active learning cycle is defined as the stage where the active learning selection strategy selects data from the unlabeled data pool and adds these data to the labeled data pool, combined with the complete network training stage. Each active learning selection strategy needs to select 60 pieces of data from the unlabeled data pool in each active learning cycle and add them to the labeled data pool to participate in the network training process of this active learning cycle. To compare fairly, an identical initial labeled data pool is needed to initialize the neural network. Therefore, in the implementation process of the comparative experiment, a random seed was specified, and 60 pieces of data were randomly selected from all unlabeled data as the initial labeled data pool. Due to the randomness of network training, even if a random seed is specified, the results of one experiment cannot represent the performance of each active learning selection strategy. Therefore, each active learning selection strategy was tested 10 times, with the same random seed for each experiment, but different random seeds for different experiments, and the average accuracy of the network model performance after each active learning cycle was taken as the average of 10 times. Comparing different active learning selection strategies in this way has stronger fairness and stability.

The experimental results of different active learning strategies are shown in Fig. 5.10, which includes the discriminative active learning selection strategy (DAL) proposed in this technical case for robot grasping, random sampling strategy (RS), graph density strategy (GD), learning loss strategy (LL), and variational adversarial active learning strategy (VAAL). As can be seen from the experimental results, the discriminative active learning selection strategy proposed in this technical case achieves the highest performance in almost every active learning cycle. Although

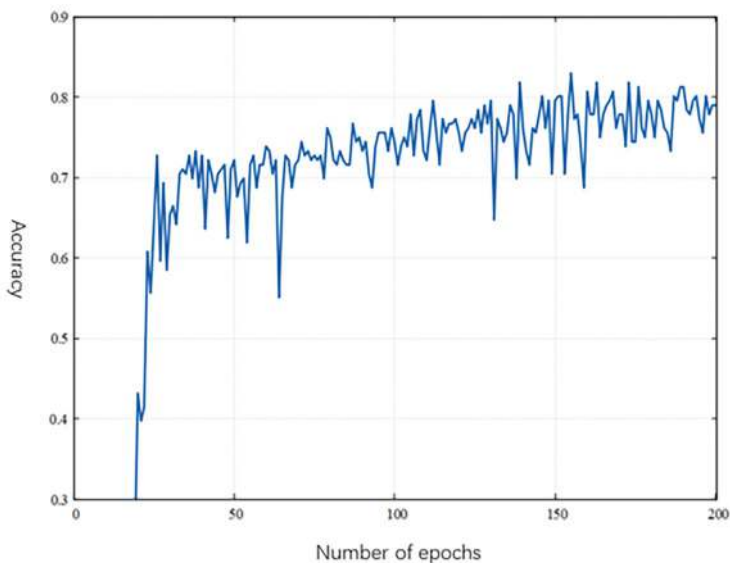


Fig. 5.9 Results of the original GG-CNN on the Cornell Grasping Dataset

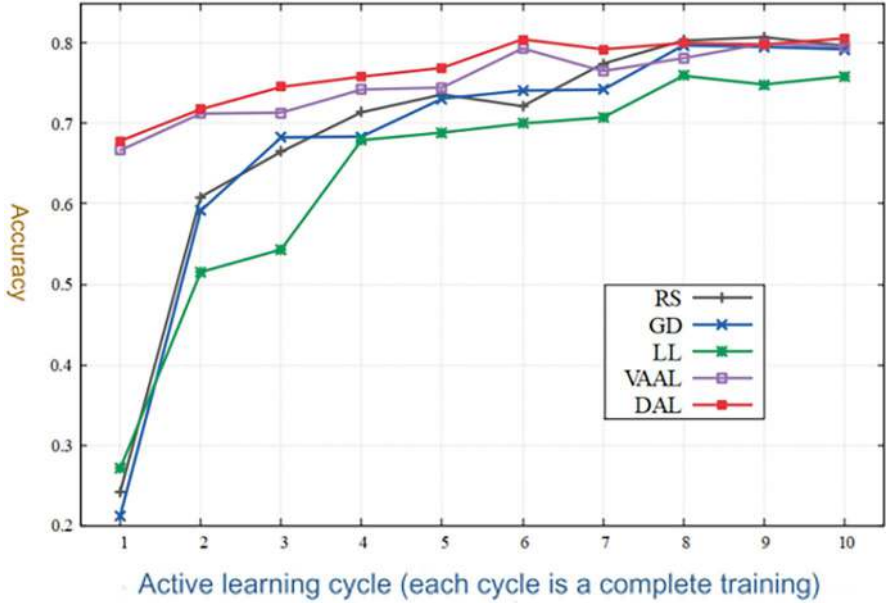


Fig. 5.10 Comparison results of Cornell grasping dataset (see color picture before the text)

the number of data in the initial labeled dataset is small, compared with random sampling strategy, graph density strategy, and learning loss strategy, this discriminative active learning selection strategy shows obvious high performance. This is due to the shared feature encoder designed in its network structure, which allows the discriminative active learning selection strategy to fully utilize the potential features of labeled and unlabeled data, thus providing more comprehensive training for the shared feature encoder. The robot grasping pose detection decoder and the active learning discriminator can also make more accurate predictions due to the fully trained shared feature encoder, and the discriminative active learning strategy can identify which data has more information, ultimately helping to improve the overall performance of the network. Compared with other active learning selection strategies, the adversarial selection strategy shows high performance, but because it ignores the naturally existing potential features of the input data provided by the robot grasping pose detection network in its network structure design, and uses an independent reconstruction network to extract the features of the input data, these features may not necessarily be representative for the input data of the target robot grasping pose detection network, so the performance of the discriminative active learning selection strategy for robot grasping will be superior. The performance of the graph density selection strategy is close to that of the random sampling strategy, while the learning loss selection strategy does not perform well on the Cornell grasping dataset. The reason may be that the overall quantity of the Cornell grasping dataset is small, which cannot provide sufficient training for the loss function prediction module in this active learning strategy, resulting in the data selected by this

active learning strategy not having enough information. Furthermore, under the data filtering of the discriminative active learning selection strategy for robot grasping, the accuracy of the robot grasping posture detection network on the test dataset can reach 80% in the sixth round, without needing to train the network with all the annotated data. Other active learning selection strategies, such as random selection strategy and graph density selection strategy, can make the robot grasping posture detection network reach close to 80% accuracy in the eighth round.

Results of the Jacquard Grasping Dataset

Similar to the experiments on the Cornell Grasping Dataset, this technical case first tested the detection performance of the original robot grasping posture detection network GG-CNN on the Jacquard Grasping Dataset. The Jacquard Grasping Dataset is a very large real object grasping dataset, and it is unrealistic to collect such a large amount of data for real-world grasping tasks. Therefore, this technical case created a subset of the Jacquard Grasping Dataset containing 300 objects and 1500 RGB-D data and used this subset to experiment and compare the performance of the discriminative active learning strategy proposed in this technical case and other active learning strategies. In terms of dataset division, unlike the Cornell Grasping Dataset, the Jacquard Grasping Dataset contains data groups of different objects. Therefore, in the division of training and test sets, this technical case randomly selected one of the five RGB-D data corresponding to each of the 300 objects in the Jacquard Grasping Dataset subset to form a test set of 300 RGB-D data, and the remaining 1200 RGB-D data were used as the training set.

The test performance of the original GG-CNN on the Jacquard Grasping Dataset subset constructed in this technical case is shown in Fig. 5.11. As can be seen from Fig. 5.11, the original GG-CNN is easier to fit on the Jacquard Grasping Dataset subset, and the performance has reached its peak after the initial 50 rounds of training, with the highest accuracy rate close to 85%, and there is no further improvement in performance in subsequent training.

In the comparative experiment of active learning selection strategies, one of the five RGB-D data corresponding to each object was randomly selected to form a test set of 300 RGB-D data, and the remaining 1200 RGB-D data were used as the training set. The training set was processed in the same way as the Cornell Grasping Dataset, initially considering these 1200 data as unlabeled data. In the selection of the initial annotated data pool, one of the remaining four RGB-D data from the 300 objects was randomly selected to form an initial annotated data pool of 300 RGB-D data. In each active learning cycle, each active learning strategy needs to select 100 data from the unlabeled data pool and add them to the labeled data pool for participation in the next round of network training. Each active learning strategy conducted five experiments on the Jacquard Grasping Dataset, specifying the same random seed for the same experiment, and different random seeds for different experiments. The final performance results of each active learning cycle were the average of the five experiments.

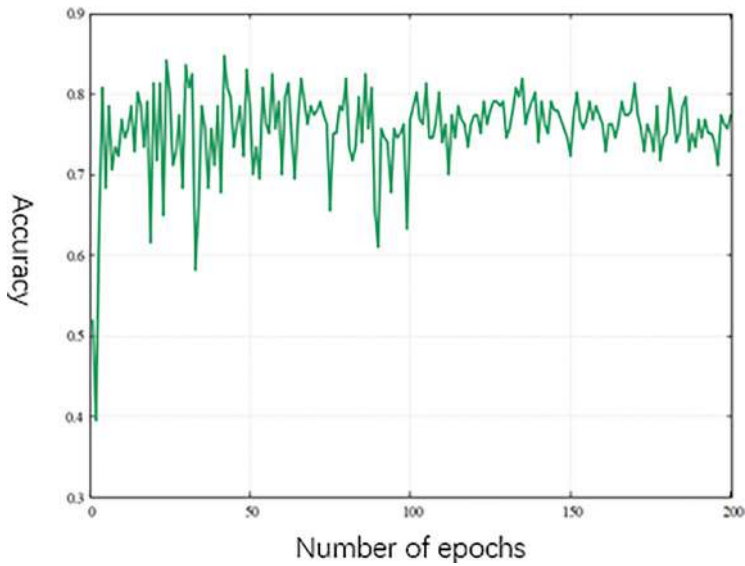


Fig. 5.11 Results of the original GG-CNN on the Jacquard Grasping Dataset subset

Figure 5.12 shows the experimental results on the Jacquard Grasping Dataset. It can be seen that in most active learning cycles, the performance of the discriminative active learning strategy for robot grasping exceeds that of other active learning strategies. In the initial stage, all active learning selection strategies have similar performance, which is different from the experimental results on the Cornell Grasping Dataset. The reason for this is that the initial annotated data pool provided in the Jacquard Grasping Dataset experiment is larger and has a richer category, while the Cornell Grasping Dataset experiment only provided a small amount of randomly selected data as the initial annotated data pool. Therefore, in the early stage of the Jacquard Grasping Dataset experiment, the robot grasping posture detection network receives more comprehensive training, resulting in similar performance results for different active learning selection strategies in the initial stage.

Furthermore, as can be seen from Fig. 5.12, the loss function selection strategy performs better on the Jacquard grasping dataset than on the Cornell grasping dataset. The reason for this is that a larger initial annotation pool provides more comprehensive training for the loss function estimation module in the loss function selection strategy. The graph density selection strategy has shown a significant decline in performance on the Jacquard grasping dataset. This may be because the data distribution characteristics of the Jacquard grasping dataset are not suitable for the graph density selection strategy to select dispersed data samples, which is a drawback of the data-based active learning selection strategy.

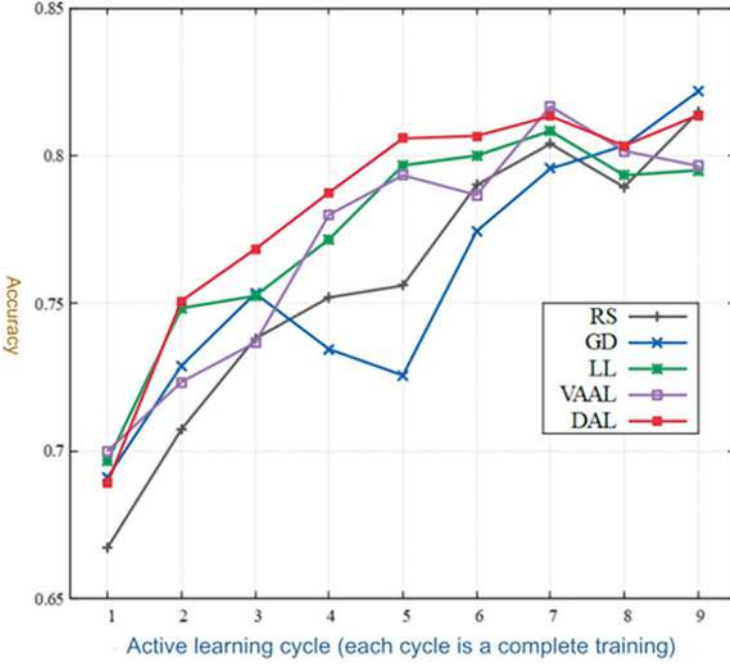


Fig. 5.12 Comparison of Jacquard grasping dataset results (see color figure before the text)

Noise Cornell Grasping Dataset Results

In real life, it is impossible to demand 100% accurate annotation information from annotators. Especially in the annotation process of robot grasping posture, annotators annotate based on experience, and there will inevitably be noise in the true values of the annotations, and the training of the grasping posture detection network is also carried out under these noisy annotations. Therefore, to simulate the situation of annotation noise in real-life grasping tasks, a Cornell grasping dataset with random noise was created using the Cornell grasping dataset, and comparative experiments of different active learning selection strategies were conducted on it.

The rotation angle in the grasping posture annotation is the most common parameter with noise. Therefore, a random rotation angle noise between -45° and 45° was added to all the rotation angles in the grasping posture annotations in the Cornell grasping dataset, to simulate a grasping dataset with annotation noise. In terms of dataset division, the same settings as in the original Cornell dataset experiment were used, with 80% of the data used for training and 20% for testing. For each active learning strategy, ten experiments were also conducted, and the final performance results were the average of the ten experiments.

Figure 5.13 shows the performance of different active learning selection strategies on the noisy Cornell grasping dataset. Although to some extent annotation noise affected all active learning selection strategies, the discriminative active



Fig. 5.13 Noisy Cornell grasping dataset experiment results (see color figure before the text)

learning selection strategy for robot grasping still showed superior performance in most active learning cycles. As mentioned earlier, the discriminative active learning strategy pays more attention to the features extracted by the shared feature encoder, and these features are also used by the robot grasping detection network, so the features extracted by the shared encoder have a stronger representativeness for the input data. In addition, the discriminative active learning strategy focuses on distinguishing which data from the unannotated data have the lowest similarity with the annotated data pool, and there are no similar data in the annotated data pool, so even if there is annotation noise, the discriminative active learning strategy can still have a certain robustness. Conversely, other active learning strategies, such as the random selection strategy, have not fully utilized the natural data features, so their performance on the noisy Cornell grasping dataset is more unstable.

5.3.2.5 Summary

The current application of intelligent robots in life is becoming more and more widespread, and autonomous grasping is an indispensable function when intelligent robots are used in reality. With the development of deep learning and computer vision, researchers have proposed many neural network models for robot grasping posture detection. However, the variability of the environment and objects still limits humans from applying robots to more scenarios. Among them, the data annotation of unfamiliar objects requires a lot of resources, including human resources

and time resources. In order to alleviate the resource consumption brought by data annotation, and to make the robot grasping detection algorithm more conveniently applied to real-life scenarios, it is necessary to use active learning strategies to screen unannotated data, and only annotate the most informative data, thereby reducing the workload of data annotation. However, the current active learning strategy faces problems such as inability to adapt and excessive parameter volume when applied to the problem of robot grasping posture detection. Therefore, this technical case proposes a discriminative active learning strategy for robot grasping, which helps the robot grasping posture detection method to select appropriate data for training. In addition, when applied to the task of robot grasping in real scenarios, the balance between the real-time performance and accuracy of the current robot posture detection algorithm is not ideal. This technical case designs a robot grasping posture detection method based on deep separable convolution, which improves the accuracy of robot grasping posture detection while taking into account real-time performance. Overall, the main achievements of this technical case are as follows:

1. A discriminative active learning strategy for robot grasping is proposed. This strategy can estimate the similarity between unannotated data samples and the annotated data set and find the unannotated data that is most dissimilar to all annotated data, that is, the most informative data, thereby helping the robot grasping posture detection network to quickly improve performance. The discriminative active learning strategy proposed in this technical case is mainly divided into three modules, including a shared feature encoder, whose main function is to extract features from the input data. The shared feature encoder can reduce the parameter volume of the discriminative active learning network, and the discriminative active learning strategy can use the same data features used during grasping posture detection, making its prediction results of the similarity between the input data and the annotated data set more accurate. There are also two structures for outputting results, one is the discriminator of the discriminative active learning strategy, which outputs the similarity between the unannotated data and the annotated data set, and the other is the decoder of the grasping posture detection network, which outputs the final grasping posture parameters. Experiments on two real object-grasping datasets and a noise-grasping dataset show that the discriminative active learning strategy can effectively select informative data, and it also has certain robustness to annotation noise.
2. A robot grasping pose detection model based on depthwise separable convolution has been proposed. This model deepens the network layers of robot grasping pose detection and increases the number of channels of the feature map, thereby improving the accuracy of grasping pose detection. The depthwise separable convolution used can limit the number of parameters, speed up the forward propagation of the network, and ensure the real-time performance of grasping pose detection. In addition, a discriminative active learning strategy is combined with this grasping pose detection model, and a complete framework from training to application of the robot grasping pose detection network is designed, making the

application of the grasping pose detection network model more convenient in real life. Comparative experiments on the accuracy of grasping pose detection were conducted on two real object grasping datasets, showing that the improved robot grasping pose detection network can better balance the real-time performance and accuracy of grasping detection. In addition, grasping experiments were also conducted on the real robot KinovaMOVO, testing the success rate of grasping on multiple objects by different network models and network models trained using the robot grasping framework. The experimental results show that the improved grasping pose detection model is superior to other grasping pose detection models, and confirm the practical application value of the grasping pose detection model training application framework designed in this technical case. Finally, the real-time detection results of the grasping pose detection network model are visualized, showing that the grasping pose detection model based on depthwise separable convolution has accurate detection results on different objects.

The active learning strategy and grasping pose detection model proposed in this technical case still have shortcomings. The discriminative active learning strategy ignores the distribution characteristics of the data when selecting informative data, which may lead to the selection of redundant data. For the grasping pose detection model, under the same network layers and feature dimensions, the performance of depthwise separable convolution is lower than traditional convolution, the model's expressive ability needs to be improved, and there is still room for improvement in the accuracy of grasping pose detection. In future work, the data distribution characteristics can be combined to help the discriminative active learning strategy select more informative data, and for the grasping pose detection model, more advanced feature extraction methods can be explored, so that more rich features can be extracted from the network while ensuring real-time performance, thereby improving the accuracy of grasping pose detection.

5.3.3 Case Study: Robot Grasping Based on Faster R-CNN

5.3.3.1 Task Description

Technical Case Three Related Code This technical case aims to solve the problem of grasping daily necessities in daily life scenes, using the Kinect v2 camera as the main sensor, processing and expanding the Cornell grasping dataset, and training the Faster R-CNN multi-layer neural network. This neural network takes RGB image files as input, predicts the reasonable grasping position of the target object, and outputs the grasping prediction rectangle of the object on the image. Subsequently, this network is migrated to the real environment in this technical case and the grasping task is completed using the KinovaMOVO robot. The main work is as follows:

Reprocessing of the Dataset

By cropping, rotating, and translating the image data in the Cornell grasping dataset in the horizontal and vertical directions, the original small-capacity dataset is expanded to the data quantity required for model training.

Grasping Rectangle Prediction Based on Convolutional Neural Network

The Faster R-CNN model was implemented, and the network structure of the model was readjusted according to the experimental environment. The adjusted model was trained using the expanded grasping dataset, and the grasping prediction box was given based on the input image information of the object to be grasped.

Transferring Experimental Results to Real-World Applications

To solve the grasping problem in real life, this technical case builds an experimental environment based on Kinect v2 and Kinova robot and completes the hand-eye calibration of the robot. Then, it detects and recognizes objects on the desktop and controls the Kinova robot through ROS to complete the grasping behavior.



Chapter Knowledge Points

Convolutional layer: Each convolutional layer in a convolutional neural network is composed of several convolutional units, and the parameters of each convolutional unit are optimized by the backpropagation algorithm. The purpose of convolution operation is to extract different features of the input. The first layer of convolution may only extract some low-level features such as edges, lines, and corners, and more layers of the network can iteratively extract more complex features from low-level features.

RPN layer (region proposal network): A candidate box extraction network, in the Fast R-CNN structure, is specifically used to extract candidate boxes, which is faster than the traditional method Selective Search.

ROI pooling layer (region of Interest): Its function is to extract the same size feature map from different sizes of RoI mapped on the convolutional feature map. Each RoI has four pieces of information (r, c, h, w) when input, where r and c represent the vertex on the top left corner of the RoI grid area to characterize the position coordinates, and h and w represent the length and width of the RoI grid area, respectively.

R-CNN network (region-CNN): Following the traditional object detection approach, it also uses extraction boxes, extracts features for each box, image classification, and non-maximum suppression in four steps for object detection. However, in the feature extraction step, the traditional features (such as SIFT, HOG features, etc.) are replaced with features extracted by deep convolutional networks.

5.3.3.2 Related Work

Object Recognition Technology

A key issue in the planning of robot grasping posture is the problem of object recognition. The recognition problem is to let the computer take a picture containing the target object as input and recognize the target object in the image. Typical object recognition algorithms can detect and describe local features in images. The algorithm finds extreme points in the difference of Gaussian (DoG) space scale and extracts its scale, position, rotation invariants, etc. Although this algorithm has the advantages of good uniqueness and rich information, it has the problems of long feature extraction time and slow speed due to its high vector dimension. A more widely used algorithm is the oriented fast and rotated brief (ORB), which uses the features from accelerated segment test (FAST) to detect feature points. The FAST algorithm selects pixels that are different from neighboring area pixels by comparing them one by one. After finding the feature points, it uses the binary robust independent elementary features (BRIEF) algorithm, which selects N pairs of points in a certain pattern around the key points, combines the comparison results of these N pairs of points, and then uses the combined binary data as the sub-description of the feature points. This algorithm is much faster than the previously proposed scale-invariant feature transform (SIFT) algorithm.

Robot Grasping Combined with Object Recognition

In the past decade, two changes have led to improvements in grasping recognition methods: the application of low-cost depth cameras and the emergence of computational frameworks that facilitate the construction and training of convolutional neural network models. The advent of consumer-grade depth cameras has enabled grasping models to encode richer features. In 2011, the work of Jiang et al. [32] represented the grasping behavior as a two-dimensional directional matrix, and their defined vector representation method is now widely adopted in the industry. Generally speaking, methods before this typically used depth cameras to perform 3D geometric reconstruction on point clouds. Since then, deep learning methods have been widely adopted in the field of computer vision, avoiding the need for engineering feature space, but at the same time, larger datasets have brought new overheads. This overhead is usually reduced by pre-training on existing datasets in the field of computer vision and fine-tuning on small datasets for specific problems. Similarly, the work of Wang and Li et al. [15] in 2016 first used an image-wide preprocessing step to identify candidate regions and then applied a convolutional neural network to each candidate region for classification. Because the grasping space is a non-convex space, regression-based methods need to be compensated by image segmentation. In 2017, the work of Kumra and Kanan et al. [33] used a two-stage neural network, the first stage outputs a feature that has been learned, and the second stage uses this feature to output a grasping behavior. In 2016, Johns et al. [34] adopted a new type of neural network, which predicts the score of each

grasping behavior in a discrete set containing all possible grasping behaviors. However, most of the deep learning methods mentioned above are based on an assumption: each image contains and only contains one object to be grasped. But this assumption is often not valid under real conditions: most objects have multiple reasonable grasping positions, and a scene often contains multiple graspable objects. In addition, reinforcement learning methods still have a major drawback: they cannot cope well with changes in the environment. There is still no good solution for how to transfer the model trained by reinforcement learning to a new unfamiliar environment.

5.3.3.3 Technical Methods

Grasping rectangle prediction method based on Faster R-CNN.

Problem Statement

Given an RGB image containing unfamiliar objects, the goal of the model is to provide feasible grasping posture predictions. Jiang et al. [32] used a seven-dimensional vector to represent the features of the grasping rectangle. This technical case uses a simplified five-dimensional matrix vector to represent the features of the grasping rectangle. The matrix g in formula (5.2) uses (x, y) to describe the coordinate position of the center of the matrix, h to represent the opening width of the parallel gripper, the angle θ to represent the direction of the matrix, and an additional global parameter w to represent the length setting of the rectangular frame.

$$g = \{x, y, \theta, w, h\}^T \quad (5.2)$$

The first three parameters of this matrix vector represent the SE(2) frame of the predicted grasping rectangle, and the last two parameters describe the dimensions of the grasping rectangle.

Introduction to Faster R-CNN Network

In this technical case, the object grasping frame prediction algorithm mainly follows the grasping rectangle frame prediction neural network proposed by Chu et al. [35] and adjusts the network structure based on this network, which mainly follows the Faster R-CNN [36] network structure.

The network model of Faster R-CNN is shown in Fig. 5.14, and the entire network is mainly divided into four parts: convolutional layer, RPN network layer, ROI pooling layer, and a fully connected neural network for classification and second precise boundary regression.

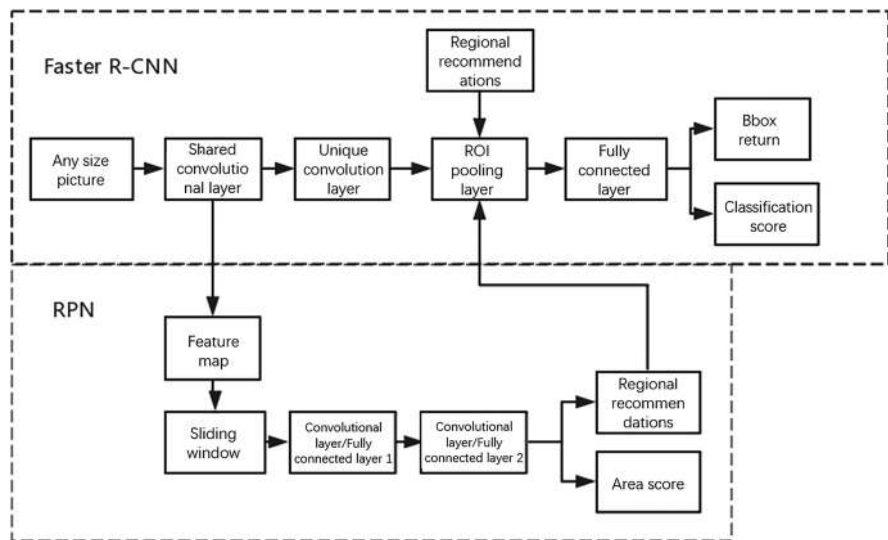


Fig. 5.14 Faster R-CNN network structure

RPN Network Layer

The RPN network is used to generate detection windows. This layer uses the normalized exponential function (SoftMax) regression to judge whether Anchors belong to the foreground or background of the image file and then uses the boundary regression method to correct Anchors to obtain precise detection windows.

The boundary regression method is used to adjust the position and size of the rectangular prediction box. A four-dimensional vector $t_i = (x, y, w, h)$ is used to represent the window position and size output by the upper network. As shown in Fig. 5.15, the goal is to train a neural network so that the blue rectangle in the figure (obtained from the output of the upper network) is mapped by the neural network to obtain a regression window G' close to the green rectangle base value (ground truth, GT), that is, the red box in the figure, given: $A = (A_x, A_y, A_w, A_h)$ and $GT = (G_x, G_y, G_w, G_h)$, find a change F , so that

$$F(A_x, A_y, A_w, A_h) = (G'_x, G'_y, G'_w, G'_h) \quad (5.3)$$

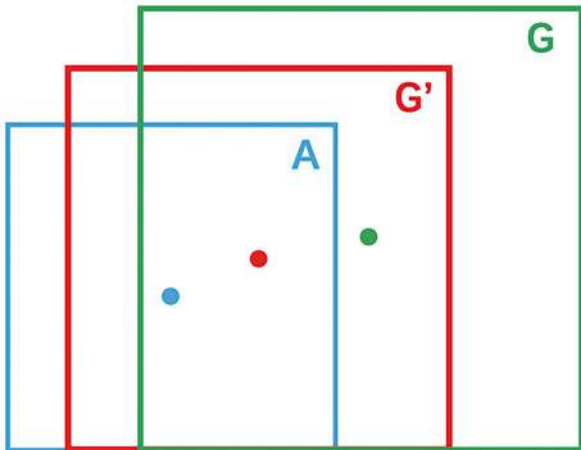
where

$$(G'_x, G'_y, G'_w, G'_h) \approx (G_x, G_y, G_w, G_h) \quad (5.4)$$

A simple change idea is to make A become G' through translation and scaling:
First, do the translation:

$$G''_x = A_w \cdot d_x(A) + A_x \quad (5.5)$$

Fig. 5.15 Schematic diagram of the first boundary regression window (see color picture before the text)



$$G_y'' = A_h \cdot d_h(A) + A_y \quad (5.6)$$

Then do the scaling:

$$G_w'' = A_w \cdot \exp(d_w(A)) \quad (5.7)$$

$$G_h'' = A_h \cdot \exp(d_h(A)) \quad (5.8)$$

So to complete the translation and scaling transformation, you need to learn the four transformations $d_x(A)$, $d_y(A)$, $d_w(A)$, $d_h(A)$. When the Anchors output by the upper network model are slightly different from GT, this change can be approximately regarded as a linear transformation, so linear regression can be used to fine-tune the position and size of the window. Therefore, the essence of the problem is to make the neural network learn a set of parameters W under the given input feature vector X , so that the value after linear regression is closer to the real value GT. For this problem, the input convolutional neural network feature map is denoted as $\varphi(A)$, and the output is the four transformations $d_x(A)$, $d_y(A)$, $d_w(A)$, $d_h(A)$. The objective function can be expressed as

$$d_*(A) = W_*^T \cdot \varphi(A) \quad (5.9)$$

where $\varphi(A)$ is the feature vector composed of the feature maps corresponding to Anchors, W_* is the parameter to be learned, and $d_*(A)$ is the predicted value obtained. According to the above, the loss function can be designed:

$$\text{Loss} = \sum_i^N \left(t_*^i - W_*^T \cdot \varphi(A^i) \right)^2 \quad (5.10)$$

The function optimization goal is

$$\widehat{W}_* = \operatorname{argmin}_{W_*} \sum_i^n \left(t_*^i - W_*^T \cdot \varphi(A^i) \right)^2 + \lambda \|W_*\|^2 \quad (5.11)$$

In Faster R-CNN, the translation amount (t_x, t_y) and scale factor (t_w, t_h) between the foreground Anchors and GT are as follows:

$$t_x = (x - x_a) / w_a \quad (5.12)$$

$$t_y = (y - y_a) / h_a \quad (5.13)$$

$$t_w = \log(w / w_a) \quad (5.14)$$

$$t_h = \log(h / h_a) \quad (5.15)$$

During the training process of boundary regression, the supervisory signal of the network is the gap between Anchors and GT. When the feature vector $\varphi(A)$ of the input feature map is given, the translation and transformation scale (t_x, t_y, t_w, t_h) of Anchors can be obtained, thus correcting the position of Anchors.

ROI Pooling Layer

This layer collects the input feature map and detection window, integrates this information to extract the proposal feature map, and sends it to the subsequent fully connected layer for target category determination.

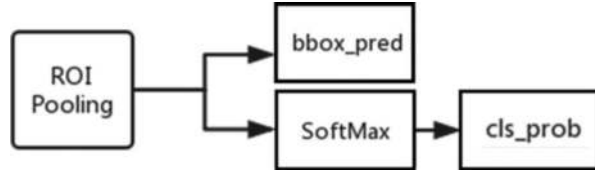
For traditional convolutional neural network models, when the network training is completed, a fixed value size image should be input, and the network output should also be a fixed size vector. When the input image size is uncertain, a part of the original image needs to be cropped and input into the network, or the image needs to be compressed to the required size and input into the network. However, cropping or compressing operations will result in the destruction of the original image structure or the loss of the original shape information of the image. Since the feature maps generated by the convolutional layer in Faster R-CNN can correspond to the original image, the $M \times N$ size detection window is mapped back to the $(M/16) \times (N/16)$ size feature map scale, and each detection window corresponding feature map area is horizontally divided into $\text{pooled\_W} \times \text{pooled\_H}$ grid, and then max pooling is performed on each part of the grid.

After such processing, the output results of different sizes of detection windows are all fixed $\text{pooled\_W} \times \text{pooled\_H}$ size, thus achieving fixed-length output.

Fully Connected Classification Neural Network

Using the detection window feature map to calculate the category of the object in the window area, and again performing boundary regression to obtain the precise location of the prediction box. The classification neural network part uses the

Fig. 5.16 Faster R-CNN fully connected layer network structure diagram



already obtained detection window feature map, calculates through the fully connected layer which specific category the image in each detection window belongs to (such as airplane, car, and TV), and outputs its corresponding probability vector; at the same time, it uses boundary regression again to obtain the position offset of each detection window, which is used to regress a more accurate target detection box. The structure of the classification part of the neural network is shown in Fig. 5.16.

5.3.3.4 Results Display

Dataset

The experiment selected the Cornell grasping dataset as the training and testing dataset for this technical case. The Cornell grasping dataset contains five types of file information: the image information of the target object in the center, the point cloud file corresponding to this image, the background image of the picture, the mapping relationship between the picture containing the target object and the background image of this picture, and the good and poor grasping frame information in these pictures. The grasping frame information uses four lines of data to describe each grasping rectangle of each picture. The four lines of data are the coordinate data of the four vertices of the rectangle, and the sample set after the rectangle is connected with yellow and blue lines is shown in Fig. 5.17.

In order to evaluate the ability of the model used in the experiment to predict grasping under multiple objects and multiple targets, a new dataset was also created for testing in addition to the Cornell grasping dataset. The following two datasets were used in this technical case:

1. Cornell grasping dataset: The Cornell grasping dataset contains a total of 885 pictures of 244 different objects. These pictures describe the corresponding objects from different directions or different poses. The configuration file of each group of pictures describes the grasping matrix data marking of the reasonable grasping method for the objects contained in the image.
2. Multi-object dataset: Because the Cornell Grasping Dataset contains only one target object per image. To achieve grasping prediction in a single image containing multiple target objects, a new grasping dataset is created according to the protocol of the Cornell Grasping Dataset. The new dataset contains 80 images, each containing more than three different target objects. Similarly, according to the Cornell dataset production protocol, a grasping prediction box annotation is made for each object in each image as a comparison baseline.

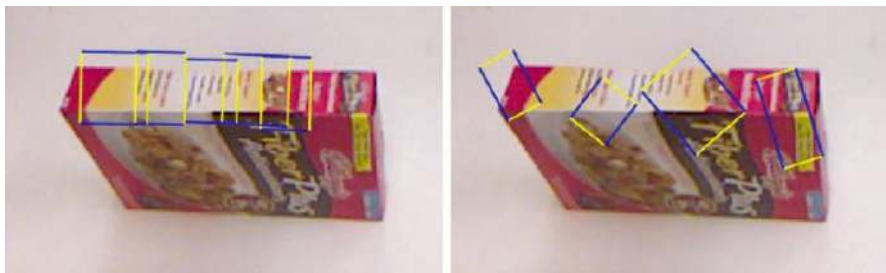


Fig. 5.17 Positive and negative prediction rectangles on the Cornell grasping dataset. (a) Display of positive prediction rectangles on the image. (b) Display of negative prediction rectangles on the image

Preprocessing and Expansion of the Dataset

To use the ResNet-101 model, which has performed well on the COCO-2014 dataset, it is necessary to preprocess the Cornell Grasping Dataset to make the preprocessed data meet the input requirements of this neural network model. At the same time, due to the small number of images in the Cornell grasping dataset, it is necessary to greatly expand its dataset. The expansion method used in this technical case is as follows: First, the images in the Cornell Grasping Dataset are cropped in the center area to obtain a 351×351 size area, then the cropped images are randomly rotated $0-360^\circ$, and the rotated images are cropped to 321×321 size. After that, the cropped images are moved $0-50$ pixels in a random direction, and finally, the images are cropped to 227×227 size to meet the input conditions of the ResNet-101 model.

Pretraining of the Model

To avoid overfitting and speed up the training process of the model, the well-trained ResNet-101 model is directly used in the network model. All networks other than ResNet are trained from scratch.

Considering the symmetry of the grasping rectangle area given in the Cornell grasping dataset, an angle θ in the range of $0-180^\circ$ can represent the tilt direction of the grasping rectangle. The experiment divides 180° evenly into R regions, each region representing a different tilt angle of the rectangle. The tilt angle of the grasping rectangle in the dataset is a continuous variable from 0 to 180° , while the tilt angle of the rectangle output by the network model in the experiment is R discrete angles. Therefore, it is necessary to pre-allocate the rectangles in the dataset to the discrete areas with the closest tilt angles.

Training

The model in this technical case uses a NvidiaTitan-V graphics card under the MaxWell architecture to train the network model. The TensorFlow framework based on cudnn-5.1.0 and cuda-8.0 is used.

Evaluation Method

The evaluation method of the experiment adopts a standard already established in the field of robot grasping. If the predicted grasping box given by the model is Gp, and the rectangle box annotated in the dataset is Gt, the experiment judges whether a rectangle's prediction is successful based on the following two conditions:

1. The angle between the predicted rectangle box Gp and the rectangle box Gt annotated in the dataset should be $<30^\circ$.
2. The Jaccard similarity score of the predicted rectangle box Gp and the rectangle box Gt annotated in the dataset should be >0.25 .

$$J(Gp, Gt) = \frac{|Gp \cap Gt|}{|Gp \cup Gt|} > 0.25 \quad (5.16)$$

Construction of the Experimental Platform

The experimental environment built in this technical case consists of an experimental host and a KinovaMOVO mobile manipulator platform equipped with a Jaco2 robotic arm, with the Kinect camera located on the head of the MOVO robot. The experimental platform is shown in Fig. 5.18. The CPU of the experimental host is Intel i9-9900k@3.6GHz×8, and the GPU is the NvidiaTitan-V graphics card under the MaxWell architecture, used for the training of the grasping box prediction neural network model. A table of appropriate height is placed in front of the MOVO grasping platform for actual object grasping.

The experiment was set up in the Ubuntu 16.04 system environment with the ROS operating system, and images of the real environment were collected by the Kinect camera. After processing and calculating the collected images, the MOVO mobile manipulator platform is remotely controlled via a local area network by the experimental host to complete the grasping behavior.

On the software side, the entire system is implemented on the TensorFlow deep learning framework and MATLAB. The manipulation of the Jaco2 robotic arm uses the MoveIt module in ROS and the relevant API functions of the Kinova robot. The TensorFlow framework was used to build and train the grasping frame prediction neural network. During the grasping experiment, MATLAB is used to process the image data input from Kinect, and the processed image file is used as the input to the grasping frame prediction neural network model on the experimental host. After

Fig. 5.18 Physical image of the visual grasping experiment platform

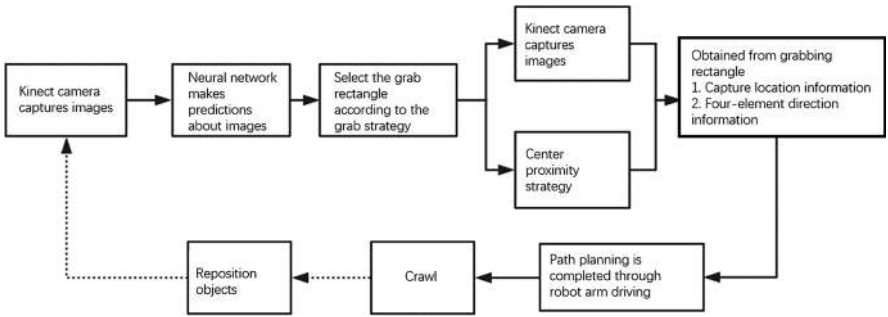


Fig. 5.19 Grasping experiment flow

calculation, all the grasping frames of the object to be grasped and the corresponding scores of each grasping frame are obtained, and the final grasping rectangle to be executed by the robot is selected according to different grasping frame selection strategies. The parameters of the final grasping rectangle output by the neural network are converted into the position information of the target object and the direction information represented by the quaternion according to the mapping rules, and the information is published to the ROS node. After the Kinova robot receives the subscribed node information, it uses its corresponding MoveIt module and driving environment to complete the enabling of the gripper, the inverse kinematics solution of the robotic arm, and the motion path planning process of the robotic arm. The experimental flow chart of the entire grasping process is shown in Fig. 5.19.



Fig. 5.20 Grasping frame prediction results under single item. (a) Original image of the item taken by Kinect V2; (b) Grasping prediction rectangle without marked score; (c) Grasping prediction rectangle with marked score

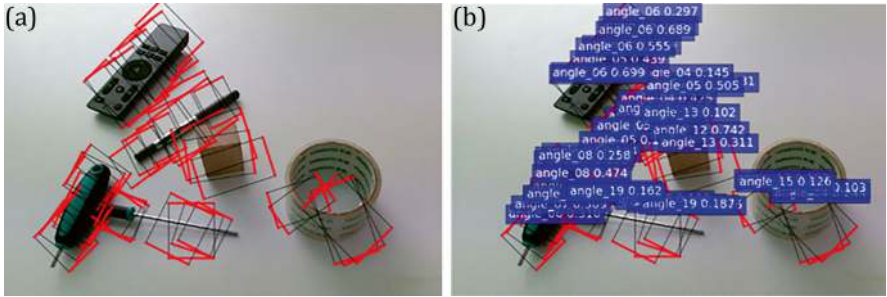


Fig. 5.21 Grasping frame prediction results under multiple items. (a) Multi-object grasping frame prediction without marked score; (b) Multi-object grasping frame prediction with marked score

The neural network implemented in this practice predicts the grasping frame of a single object as shown in Fig. 5.20. In order to more clearly represent the predicted position of the grasping frame, Fig. 5.20b removes the scores for each generated prediction frame.

For inputs containing multiple target objects, the output of the grasping frame prediction neural network is shown in Fig. 5.21.

Using the given evaluation method, tests were conducted on the Cornell grasping dataset, and the test results are shown in Table 5.8.

Image-wise and object-wise are two ways of dividing image datasets. The image-wise division method randomly divides all images into five image sets. For the grasp box prediction network model, this dataset division method can test the network

Table 5.8 Grasping frame prediction accuracy under different Jaccard indices

Image segmentation methods	Jaccard = 0.25	Jaccard = 0.30	Jaccard = 0.35	Jaccard = 0.40
Image-wise	94.7	92.5	89.8	81.4
Object-wise	96.5	91.1	86.6	80.3

Table 5.9 Grasp box prediction accuracy

Methods	Image-wise	Object-wise
Guo et al. [24]	93.2	89.1
Chu et al. (RGB) [35]	94.4	95.5
Ours (RGB)	94.7	96.5

model’s ability to predict the grasp position when facing objects in different positions and postures that it has seen before. The Object-wise division method randomly divides all object images but puts all images of one object in one image set. This division method can test the network model’s ability to predict the rectangle box of the grasp position when facing a new object it has never seen before.

Using the above evaluation methods, tests were conducted on the Cornell grasping dataset. When the Jaccard index is set to 0.25, the comparison test results with Guo [24] and Chu [35] are shown in Table 5.9.

As can be seen from Table 5.9, the network model implemented in this technical case basically reproduces the accuracy of the network model designed by Chu et al. [35]. At the same time, due to the use of the deeper ResNet-101 model, the accuracy is better than Chu’s model under the Object-wise image division method. In the hardware environment of this technical case, the average time to process an image file is about 0.18 s, which is about 5.6 fps processing speed. This processing speed has reached the standard for real-time grasping prediction problems.

A complete grasping process in a real environment is shown in Fig. 5.22. The KinovaMOVO identifies the target to be detected through its KinectV2 camera on its head and inputs the image into the grasp box prediction neural network to generate prediction results. Then, according to the generated prediction results, the robot is controlled to complete the object’s grasp through mapping rules.

5.3.3.5 Summary

This technical case implements the robot’s visual grasp of objects based on RGB images. First, a grasp prediction rectangle box generation network was researched, reproduced, and improved, and then the feasibility of the model in a real environment was verified. The main research work includes the following aspects:

1. A grasp box prediction neural network model based on convolutional neural networks was implemented and optimized. The Cornell grasping dataset was processed and expanded. The dataset was randomly divided into training and testing datasets in two different ways. The rectangle box prediction success rate of the model on the test dataset is 94.7% and 96.5% under the two different divi-

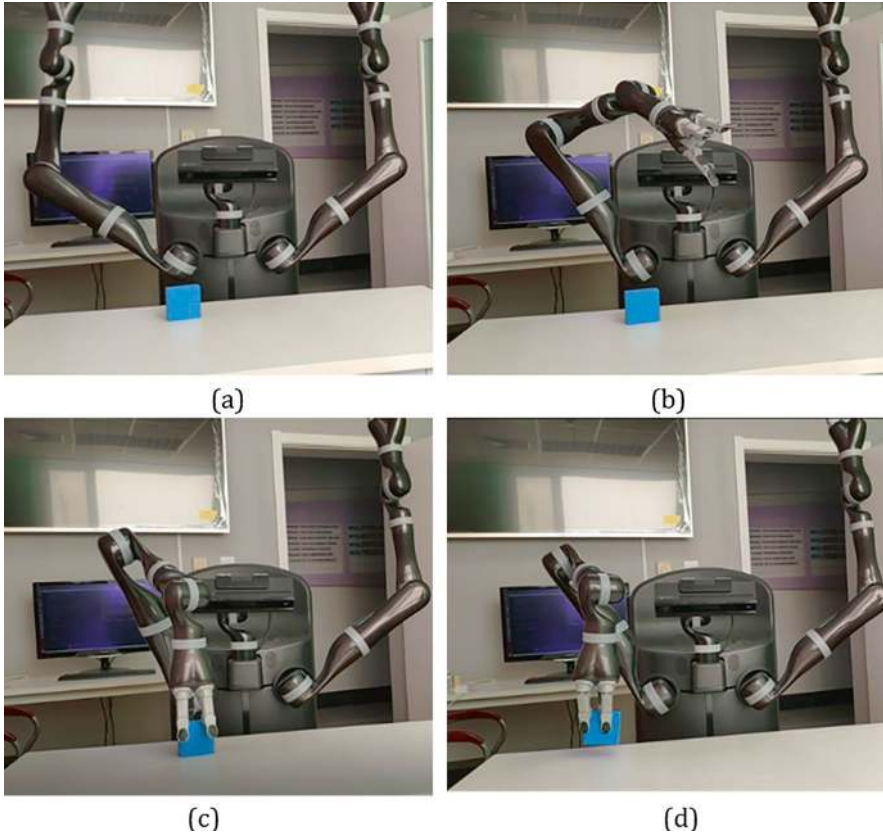


Fig. 5.22 Grasping process in a real environment. (a) Recognition; (b) approach; (c) grasp; (d) lift

sion methods, respectively. The grasp rectangle box prediction accuracy is high and can be used for grasp position prediction in a real environment.

2. Using the KinovaMOVO robot equipped with the Jaco2 robotic arm and Kinect The v2 camera sets up an experimental platform to verify the feasibility of the model. First, using different point groups in space and the least squares method, the transformation matrix from the camera coordinate system to the robot coordinate system is obtained, completing the hand-eye calibration of the robot. Then, according to the mapping rules, the position parameters represented by the rectangular frame are mapped to the grasping parameters of the robotic arm. Finally, ten everyday items are randomly placed on the table in different positions and directions five times for grasping experiments.

This technical case has conducted a certain degree of research on object grasping, achieving a high success rate and processing speed.

5.3.4 Case Study: Robot Grasping Based on Weakly Supervised Semantic Segmentation Network

5.3.4.1 Task Description

The main purpose of this technical case is to build a robot grasping system based on deep learning, find a method to synthesize a large amount of simulation data on the simulation platform, apply domain adaptive methods to make the network trained on simulation data can generalize to real scenes to give the algorithm a certain practical application value. Use unsupervised semantic segmentation methods to segment images. Summarize the current robotic arm path planning algorithm and focus on the robotic arm path planning algorithm based on deep learning. The main content of this technical case is as follows:

1. In response to the situation where it is difficult to obtain a large amount of data with high-quality pose labels in a short time, use the simulation platform to synthesize a large amount of data, and use adversarial domain adaptation methods to make the network can generalize to real scenes.
2. Design the pose estimation network architecture. The pose estimation network architecture consists of a simulation data generation module, a domain adaptation module, a pose estimation module, and a weakly supervised semantic segmentation module. Each module can be replaced with a better-performing module without affecting the performance of other modules, thereby increasing the accuracy of the entire system's pose estimation.
3. Based on the traditional path planning algorithm, adopt the path planning algorithm based on deep learning, mask and replace the pre-training network (masked and permuted pre-training net, MPNet), this planning algorithm can not only shorten the path planning time but also increase the success rate of planning based on past planning experience.



Chapter Knowledge Points

Pose Estimation: The problem of pose estimation is to determine the orientation of a 3D target object. Pose estimation is applied in many fields such as robot vision, motion tracking, and single-camera calibration. The sensors used for pose estimation in different fields are different.

Image Segmentation: Image segmentation is the technology and process of dividing an image into several specific regions with unique properties and extracting objects of interest. It is a key step from image processing to image analysis. From a mathematical point of view, image segmentation is the process of dividing a digital image into non-overlapping regions. The process of image segmentation is also a labeling process, that is, pixels belonging to the same region are assigned the same number.

Path Planning: In order to ensure that all links of the robotic arm do not collide with obstacles during the movement, a path from the starting point to the end point needs to be planned. This path should be in the working space of the robotic arm and consider the range of motion of all joints of the robotic arm.

5.3.4.2 Related Work

Pose Estimation

Most studies on object pose estimation have adopted deep learning methods. The neural network of PoseCNN uses different branches to predict the center, translation matrix, and rotation matrix of the object to predict the pose of the object [37]. This method requires a large amount of annotated data from real scenarios when training the neural network, and the large amount of time required for annotation data increases the difficulty of training the network. Some other methods use image segmentation networks to segment images to locate objects in the images. Image segmentation networks can use synthetic data for training, but the difference in distribution between synthetic data and real data makes the trained network unable to be applied to real environments.

Overall, the various excellent methods in the field of object pose estimation are roughly divided into two types: One is based on template matching methods, and the other is based on model regression methods. Both methods have their advantages and disadvantages. The six-dimensional pose estimation of objects has always been a basic problem in the field of robotics. Early research on object pose estimation algorithms was based on industrial cameras. Among the more successful algorithms are a pose estimation method based on global surface features proposed by Jones [38], and some pose estimation methods based on local surface features. Subsequently, Thachasongtham [39] and others proposed a more suitable algorithm framework, by simulating the various poses of the object in two-dimensional space in advance, and through training to screen out the most stable feature points and corresponding feature descriptions under each spatial pose of the object. Finally, during online testing, these stable feature points are used for 3D pose matching. This algorithm had a certain adaptability to large angle changes and performed well in some occlusion situations at that time. In recent years, with the continuous development of depth cameras, various new types of depth sensors have brought richer information sources to machine vision, and depth cameras have triggered a research boom in object pose estimation. Hinterstoisser [39] and others directly use depth images to calculate their corresponding gradient response maps as the basis for their feature matching, achieving very good pose estimation results, and their calculation is also very real time. With the emergence and popularity of deep learning, some object pose estimation frameworks based on deep learning have also been gradually

proposed. The deep learning feature extraction framework proposed by Wohlhart and others [41] can obtain better feature representations than LINE-MOD features. Wong and others [42] proposed an algorithm framework (SegICP) based on a deep learning framework that integrates deep semantic segmentation and pose estimation, achieving robust pixel semantic segmentation and six-dimensional pose estimation of objects. However, due to the huge nature of the deep learning framework, it is difficult to achieve efficient and convenient algorithm deployment, so the research results of similar algorithms for pose estimation using deep learning frameworks are generally not very ideal.

Semantic Segmentation

The division of the image segmentation process into feature extraction and pattern classification is due to practical considerations, not theoretical reasons. Deep neural networks can automatically learn multi-layer features in data and combine feature extraction and pattern classification into one framework. With the rapid development of deep neural networks in image classification, object detection, and other fields in recent years, deep learning methods have also been widely used in semantic segmentation tasks.

In the task of image classification, convolutional neural networks repeatedly downsample to form a pyramid from local to global, extracting more abstract and multi-scale features, and achieving a certain degree of local invariance. This can result in the final output image resolution being too small, which, although it does not affect classification, can cause problems with precise positioning when applied to semantic segmentation. The fully convolutional neural network proposes to use end-to-end, pixel-to-pixel training methods. For the problem of small resolution of the final convolutional layer feature map, a skip structure is used to combine shallow layer information and deep layer information. In addition, fine-tuning on the pre-trained image classification network can achieve better segmentation results.

Path Planning

Robotic arm path planning can be divided into two categories based on known and unknown environments: One is model-based path planning, also known as offline path planning, where the model of the robotic arm and the environment are known, and the path planning is completed before the robotic arm starts to move, and the robotic arm moves along its path; The other is sensor-based path planning. The robotic arm obtains information about the external environment from sensors in real time, thereby performing real-time path planning. Among them, model-based path planning, depending on the degree to which the robotic arm is aware of the obstacles in the environment, can be further divided into two types: Knowing all obstacle

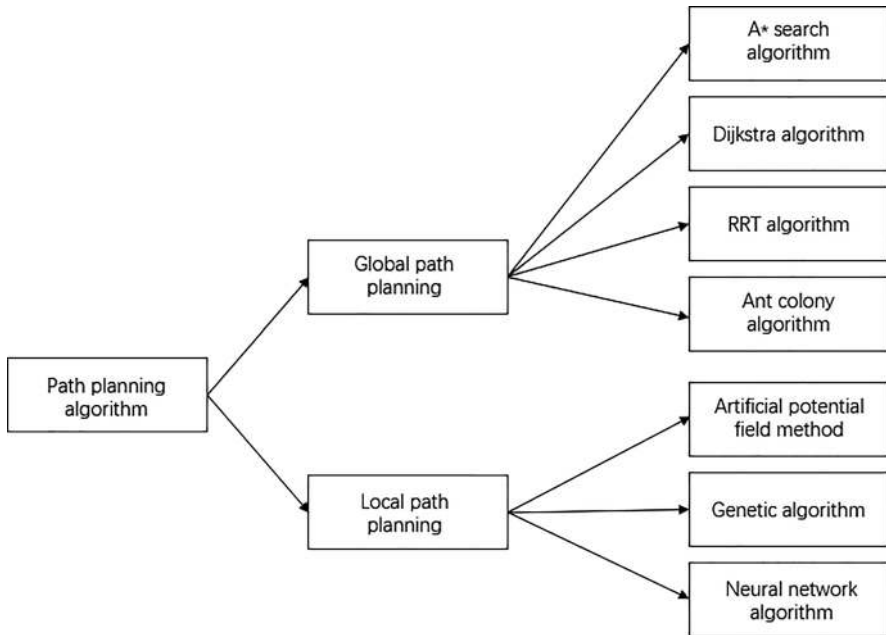


Fig. 5.23 Robotic arm path planning algorithm

information and knowing part of the obstacle information, divided into global path planning and local path planning.

The summary of the model-based path-planning algorithm for the robotic arm is shown in Fig. 5.23.

Taking the rapidly exploring random tree (RRT) algorithm as an example, the RRT algorithm [43] proposed by Lavalley in path planning has been widely used in the field of robotics, characterized by its rapid establishment of random trees. The RRT algorithm includes two parts, namely, single-RRT [43] and bi-directional RRT [44]. The heuristic node expansion method proposed by Salzman et al. [46] introduces a heuristic function, calculates the heuristic function value of the generated nodes during the process, and improves the search efficiency of RRT. Path planning first needs to set a threshold. If the heuristic value exceeds this threshold during the search process, this point is added to the random tree. Karaman et al. [47] proposed a nested RRT algorithm, which has good performance in path planning for redundant robotic arms. Janson et al. [48] addressed the lack of stability and slow convergence of the RRT algorithm and proposed the fast marching tree (FMT) algorithm. Boor et al. [49] applied the RRT search algorithm to the obstacle avoidance path planning of a six-axis robotic arm.

5.3.4.3 Technical Methods

Building an Image Generation Platform

Convolutional neural networks often require a large amount of training data to achieve excellent performance. If a dataset is made for each new object, the workload will be very large. Because the training dataset needs to consider factors such as lighting and viewing angle to include the scenes of robots in actual applications, and the pose cannot be obtained by visual observation, it is difficult to label. For a non-professional, obtaining such a database will be even more difficult. The goal of this technical case is to reduce the workload of manual annotation (including detection box, segmentation, and pose annotation). Therefore, it is necessary to build a simulation platform that can generate a large number of simulation images for training with little manual intervention.

The image synthesis method used in this technical case is domain randomization technology. Domain randomization technology adds a large amount of random noise to the synthesized images to make the images generated by the simulator more realistic. The purpose of domain randomization technology is to add a large amount of random interference to the synthesized training images, so that the trained network can generalize to real scenes. Randomness is introduced in the following aspects: (1) Increase the number of interfering objects in the image; (2) Change the texture and position of all objects in the image; (3) Change the texture of the table, chair, and robot surface; (4) Change the position and angle of the camera; (5) Change the position and direction of the light source in the scene; (6) Add random noise to the image, and randomly select the texture of all objects in the synthesized image. Use the built-in renderer of the multi-joint dynamics with contact (MuJoCo) [50] physical simulation engine to render the image. Each scene randomly adds 0–10 interfering objects. The distance from the camera in the simulator to the object is the distance from the Kinect camera to the object in the actual scene, generally 80 cm, and then 1000 viewpoints are selected on average in a sphere with a radius of 80 cm as the camera's viewpoint to collect images. After obtaining rendered images with black as the background from different viewpoints, randomly select different images from the SUN dataset as the background, and obtain a large number of new synthesized images (see Fig. 5.24).

Pose Estimation Network Structure Design

The structure of the pose estimation network is shown in Fig. 5.25, which mainly consists of the weakly supervised semantic segmentation network (WILDCAT), the pose estimation network StoCS, and the multi-adversarial domain adaptation (MADA) network. During prediction, WILDCAT outputs the probability distribution map of each class of objects, and StoCS selects points based on this probability distribution and uses point pair features for pose estimation. During training, the feature extractor is trained in an adversarial manner. The adversarial network used



Fig. 5.24 Synthesized images. (a) Synthesized image 1; (b) Synthesized image 2

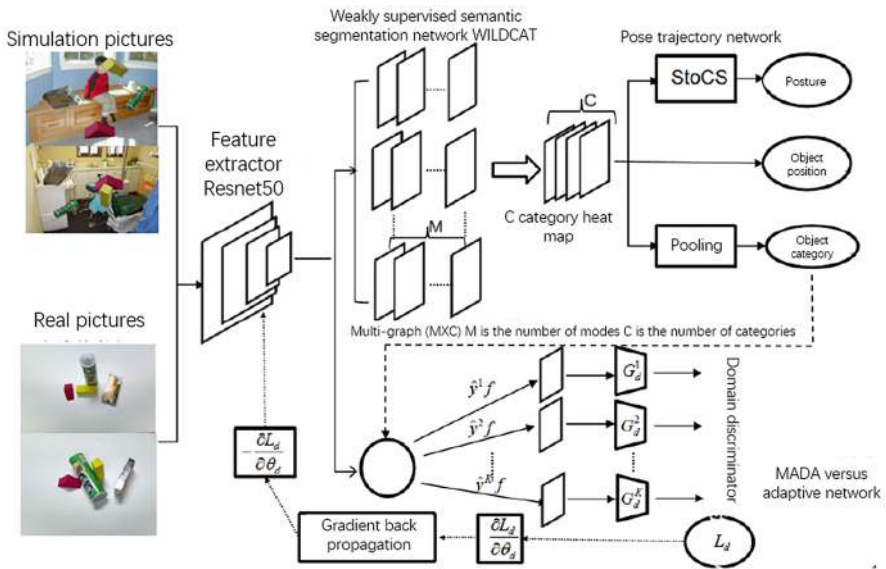


Fig. 5.25 Pose estimation network architecture

in this technical case is MADA. The following sections will detail each component module.

1. Weakly supervised semantic segmentation network WILDCAT: The network structure of the weakly supervised semantic segmentation network WILDCAT is shown in Fig. 5.26.

Weakly supervised semantic segmentation is different from supervised semantic segmentation. Weakly supervised semantic segmentation only needs the category annotation information of the image for training, and the trained network can be used for semantic segmentation of color images. When training

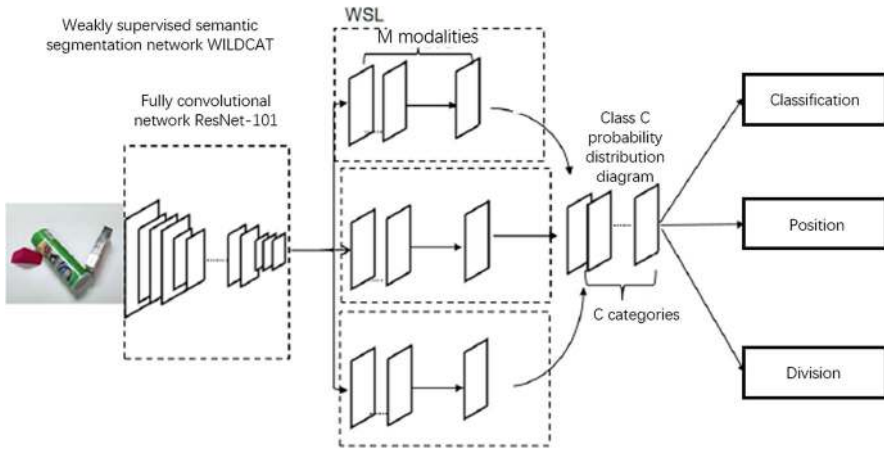


Fig. 5.26 Weakly supervised semantic segmentation network architecture

the network to recognize the object category in the image, the network will automatically learn the location information and probability distribution of each category of objects. The weakly supervised semantic segmentation network takes advantage of this feature. The network trained with category-annotated data can perform semantic segmentation on images. The image data used to train the weakly supervised semantic segmentation network are all coarsely annotated. Each image only needs to annotate how many types of objects appear in this image, and it is not necessary to annotate the category information pixel by pixel. This technical case uses the WILDCAT framework, which performs well in weakly supervised learning, as the weakly supervised learning framework. The WILDCAT framework can not only classify objects in images but also locate objects and perform semantic segmentation on images. The WILDCAT framework only needs to use image labels to train and efficiently complete three main visual tasks: image classification, object location, and semantic segmentation.

2. StoCS pose estimation algorithm: Most pose estimation algorithms first use segmentation methods to segment Red-Green-Blue (RGB) images, obtain the corresponding 3D point cloud of the segmented object based on the corresponding depth map, and finally calibrate with the actual 3D model to obtain an accurate pose. The segmentation results generally have noise, and the noise of the results is more serious when the segmentation network is trained on simulated images. Noise can make the segmentation results worse, leading to errors in the obtained 3D model, reducing the accuracy of calibration, and ultimately failing to obtain an accurate pose.

This technical case proposes a stochastic optimization process, which makes each pixel of the segmented result image represent confidence, and then tries to select areas with higher confidence as much as possible. According to the object segmentation area with high confidence, find the corresponding points in the 3D

point cloud according to the depth map. Use these points to register the 3D point cloud and the 3D model of the object in the scene to obtain the object pose. The method proposed in this technical case requires a probability map and a pre-processed set of object models. The preprocessing process is to construct a global description to describe the directional point pair features. Using this feature, not only can the sampled points be concentrated on the object boundary, but the search for the consistent point set can also be simplified. There are two advantages to using this method: first, this method does not need to obtain high-precision segmentation results, just needs to obtain the confidence probability distribution of each class after segmentation; second, combining point pair features and segmentation results to improve segmentation accuracy and improve calibration matching process will not increase computational burden. Since the segmentation network is trained on simulated data, when generalized to real data, the segmentation results will have a lot of noise, but a higher pose estimation accuracy can still be obtained.

According to the obtained RGB images, depth images, and geometric 3D models $\{M_1, M_2, \dots, M_N\}$ of N objects, estimate the poses of the known N objects $\{O_1, O_2, O_3, \dots, O_4\}$. The estimated pose is represented as a series of homogeneous coordinate transformations $\{T_1, T_2, \dots, T_N\}$, $T_i = (R_i, t_i)$, R_i and t_i , respectively, represent the rotation transformation and translation transformation relative to the camera coordinate system. Poisson sampling is performed on the computer-aided design (CAD) model of each object to convert the CAD model into a point cloud.

The process of the StoCS pose estimation algorithm is shown in Fig. 5.27.

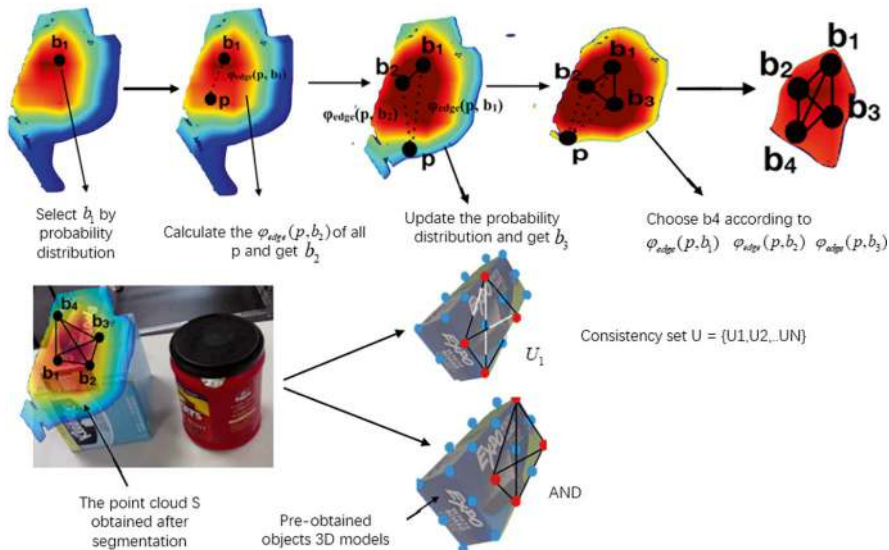


Fig. 5.27 The process of the StoCS pose estimation algorithm

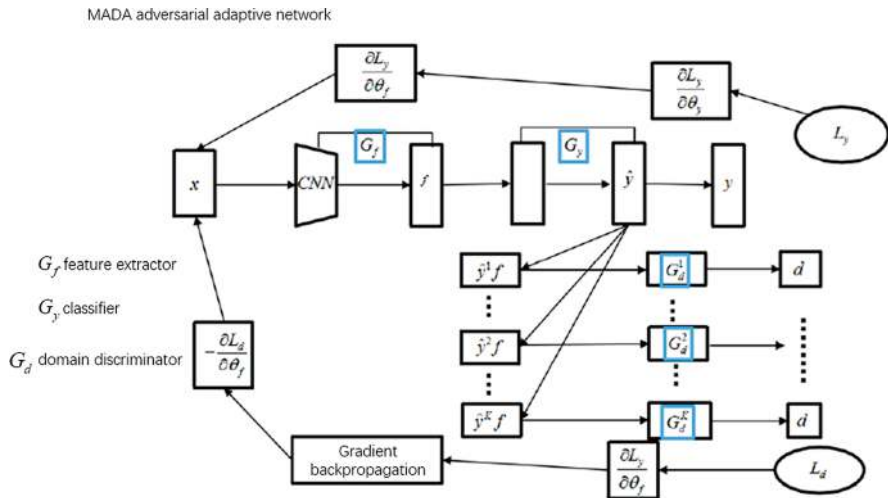


Fig. 5.28 Multimodal adversarial domain adaptive network

3. MADA network: Since the semantic segmentation model in this technical case uses a large amount of simulated data and real data for training, and the real data distribution and simulated data distribution will differ, negative transfer phenomena will inevitably occur in the process of transfer learning.

In this technical case, the MADA network is used to solve the difference between the distribution of real data and the distribution of simulated data. MADA improves the Domain Adversarial Neural Network (DANN) by adding a domain discriminator for each category, so that the probability distribution of the two inputs in the discriminator, real data and simulated data, corresponds to the same class, successfully preventing the occurrence of negative transfer phenomena in transfer learning.

The adversarial domain adaptive network structure adopted in this technical case is shown in Fig. 5.28.

The domain adaptive network reduces the data distribution difference between target domains by extracting transfer features. The adversarial learning process is a process of two players opposing each other: The first player is the domain discriminator G_d , which is trained to distinguish between the source domain and the target domain; The second player is the feature extractor G_f , which is used to extract common features from simulated images and depth images to confuse the domain discriminator. In order to extract domain invariant features f , the parameters θ_f of G_f are obtained by minimizing the loss function of the domain discriminator, and the parameters θ_d of G_d are obtained by minimizing the loss function of the domain discriminator. In addition, the parameters of the label predictor G_y can also be obtained by minimizing the loss function. After training, the parameters of G_f G_d are G_f G_d , respectively, and they will reach a saddle point.

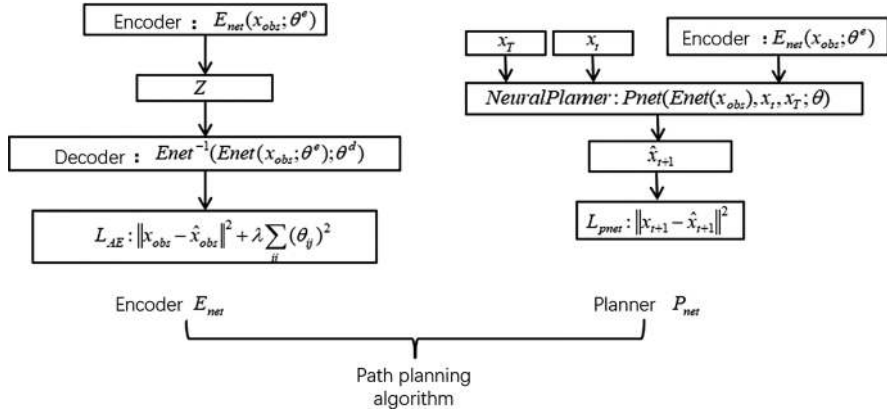


Fig. 5.29 Offline training part of MPNet

Path Planning of Robotic Arm Based on Deep Learning

The neural network-based path planning algorithm MPNet used in this technical case is divided into two parts: the first part is the offline training of the neural network; and the second part is the online generation of the planned path.

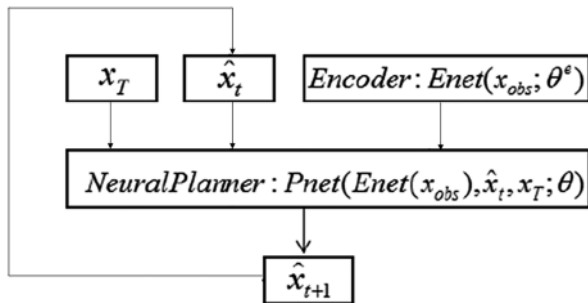
The offline part of MPNet consists of two neural network modules: the first neural network module is used to understand the entire planning scenario; the second neural network module makes plans based on the understanding of the scenario from the previous step, resulting in a point in the planned path. In this technical case, an autoencoder is used to directly encode based on the point cloud of the object, encoding the obstacle point cloud X_{obs} into a hidden space, and this network is called the encoding network (ENet). The planning network module obtains the planned path based on the understanding of the obstacles from the previous step and the start and target positions in the space. The offline training part of MPNet is shown in Fig. 5.29.

In the planning network (PNet), a feed-forward deep neural network with parameters θ is used as the planner. Based on the features Z learned from the obstacle point cloud by the encoder and the current state x_t and the target state x_T , PNet will generate the state $x_{t+1} \in X_{free}$ as the next state in the planned path, thereby guiding the robot closer to the target area. This can be described with the following mathematical formula:

$$\hat{x}_{t+1} = PNet((x_t, x_T, Z); \theta) \quad (5.17)$$

The brief flowchart of MPNet's online path planning generation is shown in Fig. 5.30. Online path planning is to explore the actual planning effect of the offline trained network in a messy and complex environment. In order to generate a path connecting the starting position and the target position end-to-end, this technical case uses an incremental bidirectional heuristic path generation algorithm. Figure 5.30 illustrates the entire process of path generation in the algorithm.

Fig. 5.30 Online path planning generation



The encoding network $\text{ENet}(x_{obs})$ obtained from offline training can encode the 3D point cloud of obstacles $x_{obs} \in X_{obs}$ into an m -dimensional feature space Z .

PNet is a feed-forward neural network trained offline, which can predict the next state $\hat{x}_{t+1}$ based on the feature space Z , the starting state x_t , and the target state x_T .

Neural network planner (neural planner). This is a heuristic incremental bidirectional path generation algorithm. The input is the feature representation Z of the obstacle point cloud, the starting state, and the target state, and the output is a path connecting the two given states.

5.3.4.4 Results Display

Experimental Results and Analysis of Domain Adaptation Network

Compare the impact of domain adaptation on network performance. Two sets of experiments were designed, each comparing the effects of three different networks. The first set of experiments is classification experiments, and the three networks to be compared are the network without domain adaptation (no adaptation), the domain adversarial neural network (DANN), and the multimodal adversarial domain adaptation network (MADA). The second set of experiments is detection experiments, and the three networks to be compared are the same as the first set of experiments.

In the classification network, it is necessary to see whether the network can distinguish which categories of objects exist in the network. In this technical case, a threshold is used. 0.5 to determine whether this object appears in the picture. During detection, a point-by-point positioning evaluation method is used, which is a standard evaluation method for evaluating weakly supervised learning to locate objects. For each class in the object, the position in the object picture corresponding to the pixel with the highest probability in the heatmap of the class is found. If this pixel is in the object's area of interest detection box, then this object location is a good object location. The results of classification and detection are shown in Fig. 5.31.

The following two conclusions can be drawn from the figure: First, it is necessary to use an adaptation network to reduce the data difference between synthetic data and real data; Second, using a multimodal domain adaptation network will bring better results than a single-modal domain adaptation network.

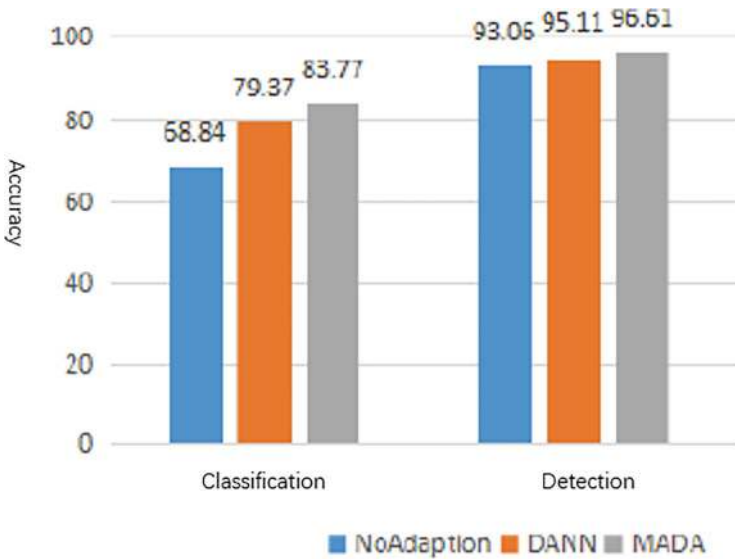


Fig. 5.31 Domain adaptation experimental results (see color figure before the text)

Weakly Supervised Network Experimental Results and Analysis

In this technical case, two sets of comparative experiments were designed to demonstrate the high accuracy of weakly supervised semantic segmentation.

The first set of comparative experiments used two training methods: training only with weakly labeled real images and training with synthetic images and fine-tuning on weakly supervised labels. The second set of comparative experiments used two training methods: training with synthetic images and fine-tuning on weakly supervised labels (MADA+fine-tuning), and training with synthetic images and weakly supervised labels (semi-supervised) at the same time (MADA+semi-supervised). The experimental results are shown in Fig. 5.32.

The results of the first set of experiments show that the performance of pre-training the network with simulated images and fine-tuning on real images is much better than training directly on real images. The second set of experiments shows that the effect of MADA+semi-supervised is slightly better than MADA+fine-tuning.

Pose Estimation Experimental Results and Analysis

The test data for this technical case comes from the test datasets in the YCB and LINEMOD datasets. The YCB dataset contains 21 classes of objects and their corresponding 3D models and has annotated the object pose and object location for 113,198 training images and 20,531 test images. YCB also provides a more challenging set of 2949 keyframe test images, and this technical case uses the 20,531 training images as the test set.

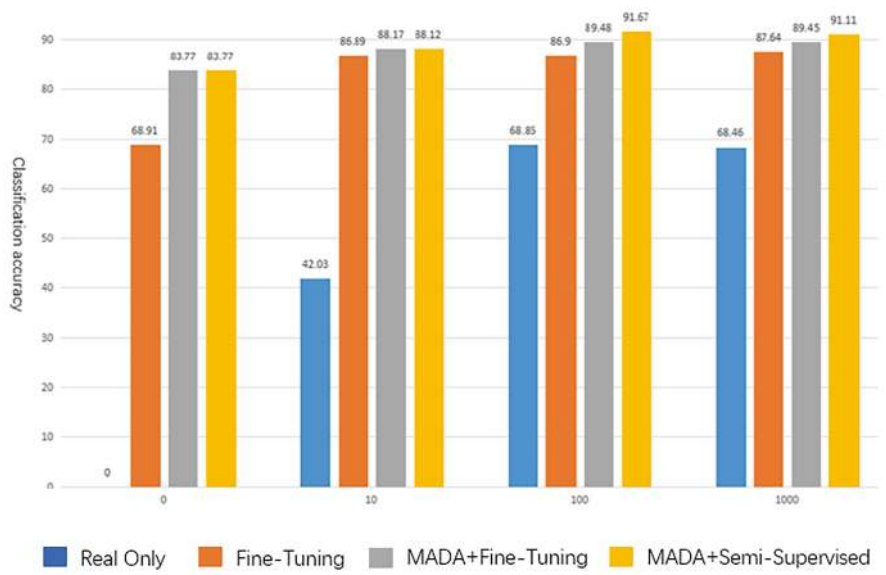


Fig. 5.32 Weakly supervised network experimental results (see color figure before the text)

Table 5.10 Accuracy of different pose estimation methods

Methods	Number of inputs	Label data	Whether to use all	Accuracy of YCB dataset	Accuracy of LINEMOD dataset
PoseCNN	RGB	Segmentation + pose	Yes	75.9	54.9
PoseCNN+ICP	RGBD	Segmentation + pose	Yes	93.0	78.0
DeepHeatmaps	RGB	Segmentation + pose	Yes	81.1	28.7
FCN+Drost	RGBD	Segmentation	Yes	84.0	—
FCN+StoCS	RGBD	Segmentation	Yes	90.1	—
Branchman	RGBD	Segmentation + pose	Yes	—	—
Michel	RGBD	Segmentation + pose	Yes	—	76.7
Ours	RGBD	object category	No	93.6	76.6

In this technical case, the network was iterated 20 times, with each iteration requiring 500 cycles. The batch size is 4, that is, four simulated images are selected, and four real images are selected. This technical case uses a learning rate of 0.001 and a momentum of 0.9 for stochastic gradient descent to optimize the network. The pre-trained Resnet50 on ImageNet is used as the feature extractor, and the parameters of the first two layers of Resnet50 are frozen.

The accuracy of different pose estimation methods is shown in Table 5.10.

The experimental results show that the pose estimation accuracy of this technical case has improved compared to previous pose estimation algorithms, but the inference time of the pose estimation algorithm is slightly longer than that of previous algorithms.

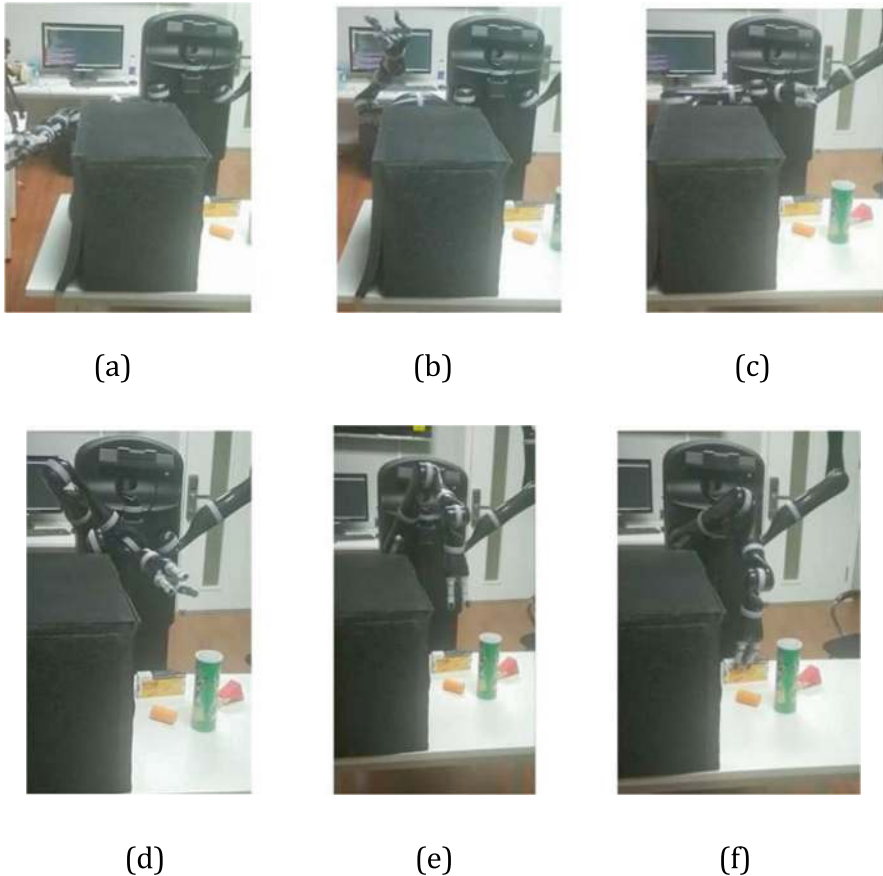


Fig. 5.33 The entire obstacle avoidance process of the Kinova robotic arm. (a) Initial pose; (b) Intermediate pose 1; (c) Intermediate pose 2; (d) Intermediate pose 3; (e) Intermediate pose 4; (f) Target pose

Results and Analysis of Robotic Arm Trajectory Planning

Experiments were conducted on the Kinova robot platform to verify whether the motion path generated by MPNet can successfully avoid obstacles, and the joint angle positions of the collision-free trajectory were sent to each motor of the Kinova robotic arm. The obstacle avoidance process is shown in Fig. 5.33. Figure 5.33a is the initial pose of the robotic arm. At this time, the end of the robotic arm is on the left side of the black obstacle, and the joint is relatively close to the obstacle, in a relatively dangerous state. The Kinova robotic arm begins to move, and the configurations at the middle 2 moments are shown in Fig. 5.33b and c. The robotic arm first moves up a distance and then moves sideways a distance. At this time, it has reached the right side of the obstacle and has successfully avoided the obstacle. Then, as shown in Fig. 5.33d and e, the position of the robotic arm is adjusted to better grasp

Table 5.11 MPNet experimental results

Test scene	New/old scene	MPNet (NR)	MPNet (HR)	Informed-R RT*	BIT*	BIT
						MPNet(NR)
Simple 2D scene	old scene	0.12 ± 0.03	0.19 ± 0.14	5.36 ± 0.34	2.71 ± 1.72	24.66
	new scene	0.12 ± 0.02	0.34 ± 0.21	5.39 ± 0.18	2.63 ± 0.75	23.89
Complex 2D scene	old scene	0.17 ± 0.06	0.61 ± 0.35	6.18 ± 1.63	3.77 ± 1.62	22.24
	new scene	0.18 ± 0.27	0.68 ± 0.41	6.31 ± 0.85	4.12 ± 1.99	22.93
Complex 3D scene	old scene	0.48 ± 0.10	0.34 ± 0.14	14.9 ± 5.39	8.57 ± 4.56	17.90
	new scene	0.44 ± 0.11	0.55 ± 0.22	15.54 ± 2.25	11.1 ± 5.61	20.22
Rigid body scene	old scene	0.32 ± 0.27	1.92 ± 1.2	30.2 ± 27.59	11.1 ± 5.59	35.01
	new scene	0.33 ± 0.13	1.98 ± 1.79	30.4 ± 12.34	11.9 ± 5.36	35.99

the object. Finally, in Fig. 5.33f, the end of the robotic arm moves to the object, closes the gripper, and grabs the object. The total planning time using MPNet is 0.4 s, successfully grabbing the object without colliding with the black obstacle.

This technical case also compares the performance of several trajectory planning algorithms, including the MPNet algorithm using the neural network planner (MPNet: NR), the MPNet algorithm using the hybrid planner (MPNet: HR), and current traditional path planning algorithms such as Informed The RRT* algorithm [51] and BIT* algorithm [52] were tested in different workspaces to evaluate their performance. These workspaces include simple two-dimensional spaces, complex two-dimensional spaces, complex 3D spaces, and rigid body spaces. Table 5.11 shows the CPU planning time of different algorithms in different workspaces, with the unit being seconds. In this context, new scenarios refer to scenes not present in the training dataset, while old scenarios refer to scenes that are present in the training dataset. The data in the table shows that, except for rigid body scenarios, the planning time of MPNet: NR and MPNet: HR is <1 s in other scenarios. The planning time of Informed RRT* and BIT* will dramatically increase with the increase of space dimensions, while the planning time of MPNet: NR and MPNet: HR is not affected by the dimension of the workspace and is relatively stable. On average, the planning speed of MPNet is 20-40 times that of Informed RRT* and BIT*.

5.3.4.5 Summary

In industrial applications, robots often encounter new objects, so it is necessary for robots to quickly adapt to new objects, that is, to train a network that can predict the pose of new objects in a short time. However, current algorithms do not have the ability to quickly adapt to new objects. This technical case mainly focuses on this issue to increase the practical value of robots.

This technical case has built a robot grasping system based on deep learning, applied a method of synthesizing a large amount of simulation data on the simulation platform, and used domain adaptation methods to generalize the network trained on simulation data to real scenes. It also uses weakly supervised semantic

segmentation to segment images. It applies a robotic arm path planning algorithm, with specific content as follows:

1. The current methods and research status of pose estimation using convolutional neural networks are summarized and analyzed, and the advantages and disadvantages of each method are analyzed.
2. A large amount of annotated data is synthesized using the simulation platform. To reduce the difference between the distribution of simulation data and real data, multimodal adversarial domain adaptation is used.
3. The research status of image synthesis platforms, domain adaptation, and weakly supervised learning at home and abroad is studied. A large amount of simulation data is synthesized using the virtual simulation platform, and this data is used to train the pose estimation network proposed in this technical case based on weak supervision and domain adaptation.
4. On the basis of traditional path planning algorithms, the deep learning-based path planning algorithm MPNet is adopted. This planning algorithm can not only shorten the path planning time but also increase the success rate of planning based on the learned planning experience.

In summary, the object pose detection algorithm proposed in this technical case has strong universality, strong generalization ability, high accuracy, and certain engineering practicality. The path planning algorithm proposed in this technical case has a high success rate and short planning time. The above two parts constitute a real-time robot grasping system.

5.3.5 Case Study: Generative-based Grasping Pose Detection

5.3.5.1 Technical Methods

Literature [57] proposes a real-time, lightweight, scalable object grasping detection algorithm, which uses a smaller network and generates a grasping pose for each pixel to solve the problem of long-time consumption and extraction of candidate grasping boxes, achieving closed-loop grasping.

Grasping Representation

As shown in Fig. 5.34, this algorithm uses a fully convolutional network, using

$$g = (P, \Phi, w, q) \quad (5.18)$$

Here, it represents grasping, the dexterous hand extends in a direction perpendicular to the x - y plane, where P is the center position of the gripper, P is represented as (x, y, z) , Φ is the deflection angle of the gripper along the z -axis, w is the width of the gripper opening, and q is the confidence of the grasp.

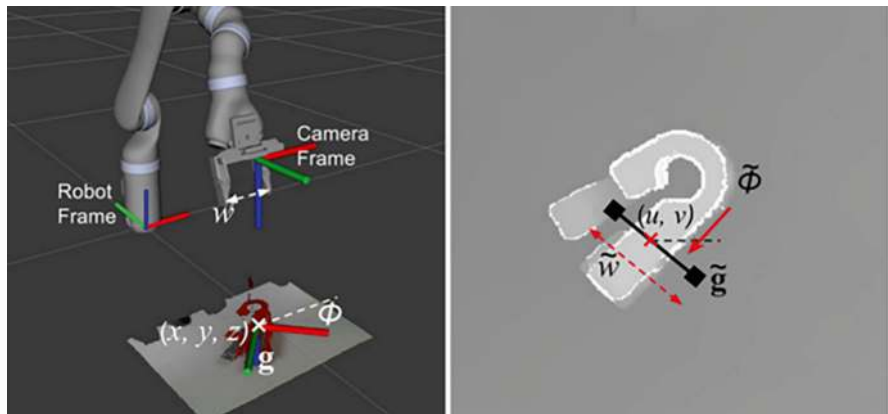


Fig. 5.34 Representation of dexterous hand grasping

The algorithm obtains the grasp pose in the plane from a depth map of size $H \times W$ (with known camera parameters):

$$\tilde{g} = (s, \tilde{\Phi}, \tilde{w}, q) \quad (5.19)$$

Here, s is the pixel coordinate of the center of the grasp, represented as (u, v) ; $\tilde{\Phi}$ is the rotation angle along the z -axis in the camera reference coordinate system; $\tilde{w}$ is the opening width of the gripper, then $\tilde{g}$ can be obtained through

$$g = t_{RC} (t_{CI} (\tilde{g})) \quad (5.20)$$

to be transformed into the real-world grasp pose g . Here, t_{CI} is the transformation matrix from the depth map plane coordinate system to the camera coordinate system, and t_{RC} is the transformation matrix from the camera coordinate system to the robot (world) coordinate system.

Since the plane coordinates of each point in the depth map are known, the next task is: Given a depth map, obtain Φ, w, q on each point. Thus, the input and output of the neural network are clear: The input is a depth map, and the output is three maps of the same size, the values on the corresponding coordinates represent the rotation angle Φ , width w , and confidence q . Finally, the Φ, w, q coordinates s at the position with the highest confidence is the optimal $\tilde{g}$, which is obtained through the above two transformation matrices, to get the optimal grasp pose g in the robot coordinate system.

Grasp Convolutional Neural Network

This neural network can approximate complex functions: $M = I \rightarrow G$,

$$M_{\theta}(I) = (Q_{\theta}, \mathcal{S}_{\theta}, W_{\theta}) \approx M(I) \quad (5.21)$$

The above equation can be trained with a set of inputs I_r , corresponding outputs G_r , and the loss function is L2, that is,

$$\theta = \arg\min_{\theta} \mathcal{L}(G_r, M_{\theta}(I_r)) \quad (5.22)$$

1. Grasp representation: G estimates the parameters of a set of grasps performed at Cartesian point p, which corresponds to each pixel coordinate s. The grasp map G is represented as a group of three images, namely, Q, Φ , and W. They are represented as follows:

Q: Describes the confidence of performing a grasp at each point (u, v). The value range is [0,1] the larger the value, the higher the success rate of the grasp.

Φ : Describes the rotation angle of the grasp performed at each point. Since the two-finger grasp is symmetrical to $\pm \frac{\pi}{2}$, the value range of Φ is $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

W: Describes the width of the gripper at each point, the range is [0,150]. Later in the actual training, all outputs will be normalized to 1, and the real value will be inferred from the prediction result, which is a common method.

2. Training dataset: To train the network, this technical example created a dataset based on the Cornell Grasping Dataset [53]. The Cornell Grasping Dataset contains RGB-D images of 885 real objects, as well as 5110 valid grasping frames and 2909 invalid grasping frames manually marked. Although it is a relatively small grasping dataset compared to some newer synthetic datasets [14, 61], it is most suitable for this study's pixel grasping representation because each image provides multiple grasping frames. This technical example uses random cropping, scaling, and rotation to expand the Cornell grasping dataset, increasing the dataset to 8840 depth images and related grasping images G_r , and 51100 grasping frames.

The Cornell grasp dataset represents two-finger grasps as rectangles in the pixel coordinate system, aligned with the position and rotation of the gripper [53]. To modify the Cornell dataset to suit the grasp representation in this practice, we extract the middle 1/3 of the rectangle annotated by Cornell, as shown in Fig. 5.35.

This method only considers the valid grasp boxes used for training the network and assumes that any grasp in any other area is invalid. The meanings of each picture are as follows:

- (a) **Confidence:** Each valid grasp box in the Cornell grasp dataset is considered as a binary label, and the corresponding area of Q_r is set to 1, with the confidence of all areas outside the rectangle set to 0.
- (b) **Rotation angle:** During training, the rotation angle Φ is encoded as two vectors $\sin(2\Phi)$ and $\cos(2\Phi)$ within the unit circle, so the predicted value is limited between $[-1,1]$, and the real rotation angle is deduced from the predicted value. The advantage of this is to eliminate the discontinuity of the data.
- (c) **Grasp width:** Calculate the grasp width (up to 150) of each pixel point of each grasp box, which represents the width of the gripper, and set the corresponding W_r . During training, the value of W_r is scaled by 1/150 to make

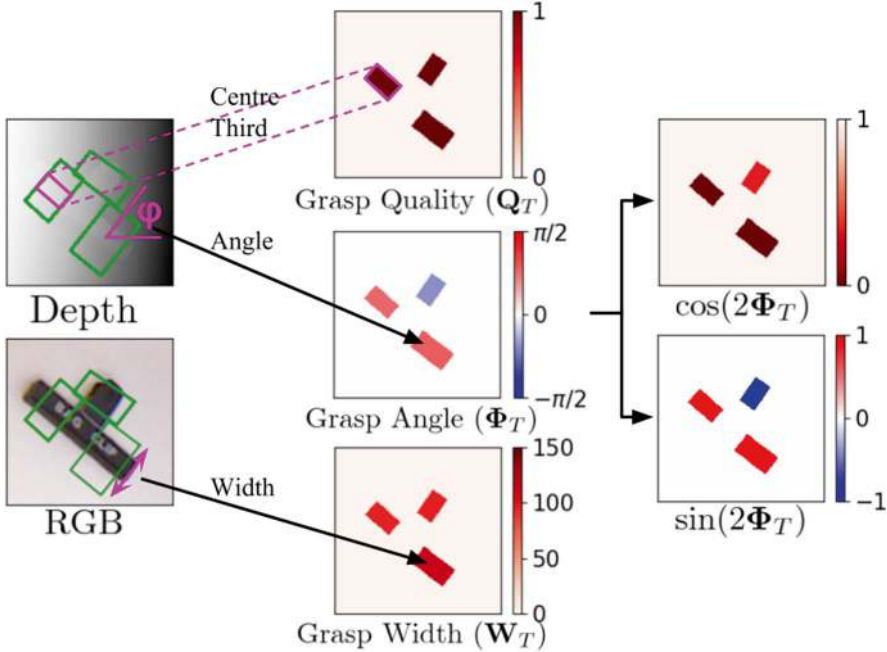


Fig. 5.35 Generation of data for training GG-CNN

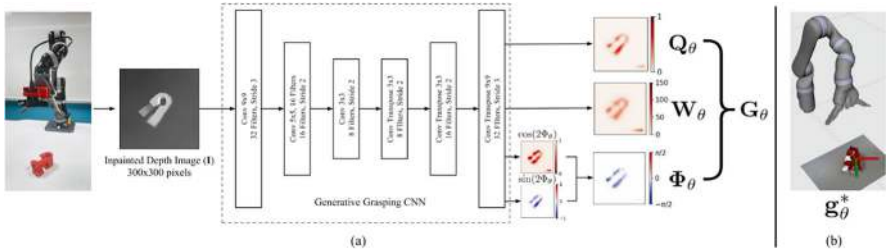


Fig. 5.36 GG-CNN structure. (a) The overall network structure of GG-CNN; (b) The best grasp point generated by GG-CNN

- it within the range $[0,1]$. The physical gripper width can be calculated using the camera's parameters and the measured depth.
- (d) **Input depth map:** Since the Cornell grasp dataset is collected using a real camera, it already contains real sensor noise, so there is no need to add noise. By using OpenCV to remove invalid depth values NaN, each depth value is subtracted from the average of all depth values in the image to obtain depth invariance with 0 as the center.
 - (e) **Network structure:** GG-CNN uses a fully convolutional network, which has been proven to perform well in computer vision tasks, such as image segmentation [61] and contour detection. As shown in Fig. 5.36.

After the last convolution layer of the network, 4 maps of the same size are output, i.e., the original angle Φ first outputs 2 maps $\sin(2\Phi)$ and $\cos(2\Phi)$, then 1 map is obtained through the inverse tangent formula, and finally a total of 3 maps.

- (f) **Training:** 80% of the dataset is used for training and 20% for validation. After designing the structure, 95 networks were obtained by changing hyperparameters such as convolution size and stride. Each network was trained for 100 epochs, and the network that performed best on the validation set was selected as the final network.

5.3.5.2 Results Display

Experimental Robot and Test Object

Experiments were conducted with the Kinova robotic arm and a two-fingered dexterous hand. The experimental grasping objects (Fig. 5.37) are divided into two groups, one group is 8 complex 3D printed parts, and the other group is 12 common items.

Grasp Detection Process

First, the depth map size is cropped to 300 pixels $\times$ 300 pixels, and invalid values are removed with OpenCV; Then it is fed into the network, and a Gaussian filter is used to filter the predicted confidence, which can remove outliers and local maxima, improving robustness. The parameters at the position with the highest confidence are selected as the optimal grasp parameters, and the grasp pose in the robot



Fig. 5.37 Experimental grasping objects

coordinate system is restored through the transformation matrix, and finally, the robot runs to achieve grasping.

Grasp Operation

The system is evaluated using two methods.

1. Open-loop grasping: The camera (on the wrist of the robotic arm) initially is 350 mm above the table and perpendicular to it. The object is placed within the camera's field of view. After calculating the optimal grasping pose, the robot's dexterous hand moves to a position 170 mm above the grasping pose, descends vertically until it reaches the grasping pose or detects a collision. The dexterous hand closes and lifts, and if it successfully returns to the starting pose, the experiment is considered successful.
2. Closed-loop grasping: The camera (on the wrist of the robotic arm) initially is 400 mm above the table and perpendicular to it. The object is placed within the camera's field of view. The depth camera takes depth images at a frequency of 30 Hz, and the grasping detection system calculates the grasping pose in real time. Since there may be multiple optimal grasping poses in one image, to avoid the system constantly switching grasping targets during the grasping process, causing system confusion, the three grasping poses with the highest confidence are selected from each depth image, and the grasping pose closest to the optimal grasping pose of the previous frame is chosen as the target for the next step. The initial optimal grasping pose is the global maximum. To make the movement faster when far from the target and slower as it approaches the target, the movement speed V is set as follows:

$$V = \lambda \left(T_{g_0} - T_f \right) \quad (5.23)$$

where T_{g_0} and T_f are the optimal grasping pose and the pose of the dexterous hand, represented by $(x, y, z, \alpha, \beta, \gamma)$ for 3D coordinates and Euler angles. λ is the speed scale, a constant.

Experimental Results

The experiment tested the grasping success rate of two groups of items in dynamic and static environments using open-loop and closed-loop grasping. The experimental results are shown in Table 5.12:

In addition to the above experiments, an imprecise motion model was designed to test the system's grasping success rate under conditions of sensor and robot control imprecision. Since the Kinova robot has high precision, to mimic imprecise control, a velocity disturbance matrix was set in the x, y, z directions:

Table 5.12 Grasping success rate under various experimental conditions

Experimental conditions	Crawl success rate
3D printed parts, static environment, open-loop gripping	84% (67/80)
3D printed parts, static environment, closed-loop gripping	81% (65/80)
3D printed parts, dynamic environments, closed-loop grasping	83% (66/80)
Common objects, static environment, open-loop grasping	92% (110/120)
Common items, static environment, closed-loop grasping	91% (109/120)
Common objects, dynamic environments, closed-loop grasping	88% (106/120)

$$V_c = V \cdot \begin{pmatrix} 1+c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & 1+c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & 1+c_{zz} \end{pmatrix}$$

where $c \sim N(0, \sigma^2)$ is the velocity disturbance ratio between different axes. After each detection of the optimal grasping pose, disturbance is added to each axis of the optimal pose. The results are shown in Fig. 5.38.

5.3.5.3 Summary

In this technical case, a generative-based grasping pose detection algorithm is introduced. This algorithm uses a smaller network and generates a grasping pose for each pixel to solve the problem of long-time consumption and complex candidate box extraction. By inputting a depth map into the network, three images of the same size are obtained, and the values on the corresponding coordinates represent rotation angle, width, and confidence, respectively. The coordinates at the position with the highest confidence are the optimal grasping pose sought, and the optimal grasping pose in the robot coordinate system can be obtained through the transformation matrix. This algorithm does not sample the candidate box, uses a fully convolutional neural network to directly output the grasping pose on each pixel end-to-end, and has fewer parameters, with a running time of 19 ms, which can fully implement closed-loop control. Experiments show that the success rate of this algorithm on complex geometric objects is 83%, the success rate on moving common items is 88%, and the success rate when multiple objects are chaotically stacked and moving is 81%. It can be seen that this algorithm still has a high success rate when the target is moving and the control of the robotic arm is not accurate enough.

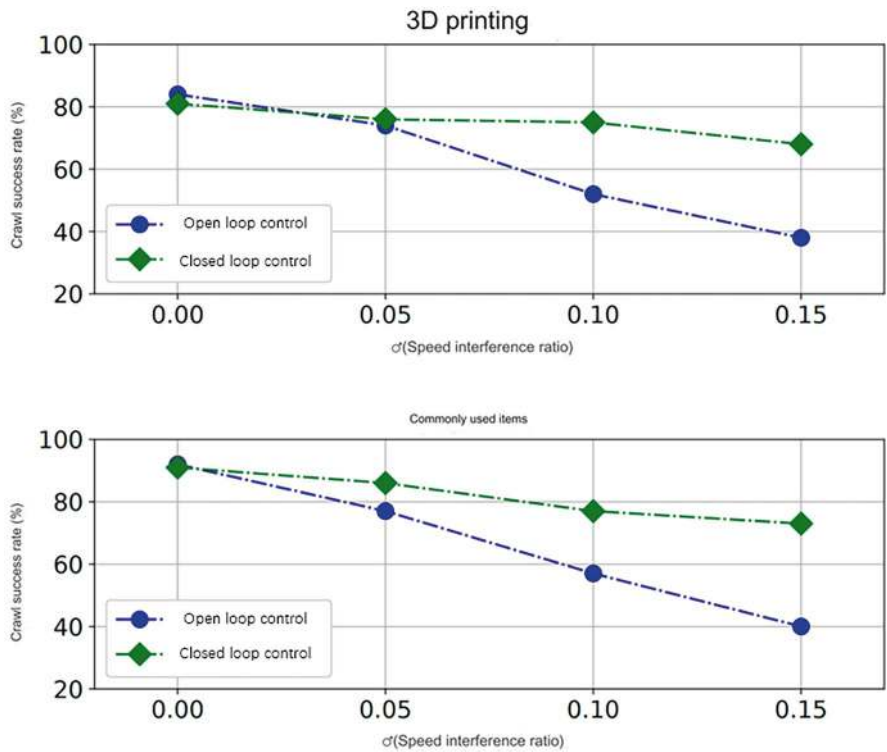


Fig. 5.38 Grasping success rate of the imprecise motion model. (a) 3D printed parts; (b) Common items

5.4 Summary

This chapter first gives a simple overview of the grasping problem in robot interaction and then introduces the domestic and foreign research status of robot grasping from three research perspectives: analytical method, empirical method, and deep learning method. It is found that traditional analytical methods and empirical methods have significant limitations, and only deep learning-based methods can perform well in actual scenarios and have strong generalization ability for unknown objects. Then, robot grasping is practiced from the above several aspects. First, based on the analytical method and empirical method, the method based on interactive dynamics is used for practice, and then following the current research hotspot, four deep learning-based grasping detection methods are proposed for discussion and practice.

5.5 Discussion

1. Please state the basic concept of grasping pose detection, what are the indicators for evaluating grasping pose, and what are the mainstream methods?

2. Please analyze the characteristics of the unstructured environment.
3. Please briefly describe the basic concept of robotic arm path planning and the role of image segmentation.
4. Please briefly describe the basic concept and function of the ROI pooling layer.
5. Please briefly describe the active learning strategy in this chapter's active learning-based grasping.
6. Practice: Based on the generative robot grasping network, manually implement a fully convolutional pose estimation network.
7. Practice: Please refer to the active learning strategy in this chapter, try to build an active learning network based on the pool, and judge the size of the information content of the unlabeled data in the dataset.

References

1. EKVALL S, KRAGIC D. Learning and evaluation of the approach vector for automatic grasp generation and planning[C]//Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007: 4715–4720.
2. MORALES A, ASFOUR T, AZAD P, et al. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands[C]//2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006: 5663–5668.
3. BICCHI A, KUMAR V. Robotic grasping and contact: A review[C]//Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat.No.00CH37065). IEEE, 2000, 1: 348–353.
4. PRATTICIZZO D, MALVEZZI M, GABICINI M, et al. On the manipulability ellipsoids of underactuated robotic hands with compliance[J]. Robotics and Autonomous Systems, 2012, 60(3): 337–346.
5. ROSALES C, SUAREZ R, GABICINI M, et al. On the synthesis of feasible and prehensile robotic grasps[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 550–556.
6. NGUYEN V D. Constructing force-closure grasps[J]. The International Journal of Robotics Research, 1988, 7(3): 3–16.
7. ROA M A, SUAREZ R. Computation of independent contact regions for grasping 3D objects[J]. IEEE Transactions on Robotics, 2009, 25(4): 839–850.
8. KRUG R, DIMITROV D, CHARUSTA K, et al. On the efficient computation of independent contact regions for force closure grasps[C]//2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2010: 586–591.
9. RODRIGUEZ A, MASON M T, FERRY S. From caging to grasping[J]. The International Journal of Robotics Research, 2012, 31(7): 886–900.
10. SEO J, KIM S, KUMAR V. Planar, bimanual, whole-arm grasping[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 3271–3277.
11. ZHANG L, TRINKLE J C. The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 3805–3812.
12. MORRISON D, CORKE P, LEITNER J. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach[J]. arXiv preprint arXiv: 1804.05172, 2018.
13. VARLEY J, DECHANT C, RICHARDSON A, et al. Shape completion enabled robotic grasping[C]//2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 2442–2447.

14. MAHLER J, LIANG J, NIYAZ S, et al. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics[J]. arXiv preprint arXiv: 1703.09312, 2017.
15. WANG Z, LI Z, WANG B, et al. Robot grasp detection using multimodal deep convolutional neural networks[C]. *Advances in Mechanical Engineering*, 2016, 8(9): 1687814016668077.
16. VARLEY J, WEISZ J, WEISS J, et al. Generating multi-fingered robotic grasps via deep learning[C]//2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2015: 4415–4420.
17. MAHLER J, POKORNY F T, HOU B, et al. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards[C]//2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016: 1957–1964.
18. ZENG A, YU K T, SONG S, et al. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge[C]//2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 1386–1383.
19. MILLER A T, ALLEN P K. Graspit! a versatile simulator for robotic grasping[J]. *IEEE Robotics & Automation Magazine*, 2004, 11(4): 110–122.
20. BOHG J, MORALES A, ASFOUR T, et al. Data-driven grasp synthesis—a survey[J]. *IEEE Transactions on robotics*, 2013, 30(2): 289–309.
21. BROOK P, CIOCARLIE M, HSIAO K. Collaborative grasp planning with multiple object representations[C]//2011 IEEE international conference on robotics and automation. IEEE, 2011: 2851–2858.
22. HINTERSTOISSER S, HOLZER S, CAGNIART C, et al. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes [C] //2011 international conference on computer vision. IEEE, 2011: 858–865.
23. TEN PAS A, PLATT R. Using geometry to detect grasp poses in 3d point clouds [J]. *Robotics Research: Volume 1*, 2018: 307–324.
24. GUO D, KONG T, SUN F, et al. Object discovery and grasp detection with a shared convolutional neural network [C] //2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016: 2038–2043.
25. CHEN X, MA H, WAN J, et al. Multi-view 3d object detection network for autonomous driving [C] //Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017: 1907–1915.
26. SONG S, XIAO J. Deep sliding shapes for amodal 3d object detection in rgb-d images [C] //Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 808–816.
27. ZHOU Y, TUZEL O. Voxelnet: End-to-end learning for point cloud based 3D object detection [C] //Proceedings of the IEEE conference on computer vision and pattern recognition, 2018: 4490–4499.
28. LI J, CHEN B M, LEE G H. So-net: Self-organizing network for point cloud analysis [C] //Proceedings of the IEEE conference on computer vision and pattern recognition, 2018: 9397–9406.
29. TEN PAS A, GUALTIERI M, SAENKO K, et al. Grasp pose detection in point clouds [J]. *The International Journal of Robotics Research*, 2017, 36(13–14): 1455–1473.
30. YAN X, HSU J, KHANSARI M, et al. Learning 6-dof grasping interaction via deep geometry-aware 3d representations [C] //2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 3766–3773.
31. QI C R, SU H, MO K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation [C] //Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 652–660.
32. JIANG Y, MOSESON S, SAXENA A. Efficient grasping from rgb-d images: Learning using a new rectangle representation [C] //2011 IEEE International conference on robotics and automation. IEEE, 2011: 3304–3311.
33. KUMRA S, KANAN C. Robotic grasp detection using deep convolutional neural networks [C] //2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 769–776.

34. JOHNS E, LEUTENEGGER S, DAVISON A J. Deep learning a grasp function for grasping under gripper pose uncertainty [C] //2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016: 4461–4468.
35. CHU F J, XU R, VELA P A. Real-world multiobject, multigrasp detection [J]. IEEE Robotics and Automation Letters, 2018, 3(4): 3355–3362.
36. REN S, HE K, GIRSHICK R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks [J]. Advances in neural information processing systems, 2015, 28.
37. XIANG Y, SCHMIDT T, NARAYANAN V, et al. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes [J]. arXiv preprint arXiv: 1711.00199, 2017.
38. ASTHANA A, MARKS T K, JONES M J, et al. Fully automatic pose-invariant face recognition via 3D pose normalization [C] //2011 International Conference on Computer Vision. IEEE, 2011: 937–944.
39. THACHASONGTHAM D, YOSHIDA T, DESORBIER F, et al. 3D object pose estimation using viewpoint generative learning [C] //Image Analysis: 18th Scandinavian Conference, SCIA 2013, Espoo, Finland, June 17–20, 2013. Proceedings 18. Springer Berlin Heidelberg, 2013: 512–521.
40. WOHLHART P, LEPETIT V. Learning descriptors for object recognition and 3d pose estimation [C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 3109–3118.
41. WONG J M, KEE V, LE T, et al. Segicp: Integrated deep semantic segmentation and pose estimation [C] //2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 5784–5789.
42. YAN R J, WU J, LEE J Y. Motion planning of unicycle-like robot using single RRT with branch and bound algorithm [C] //2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). IEEE, 2013: 599–601.
43. MARTIN S R, WRIGHT S E, SHEPPARD J W. Offline and online evolutionary bi-directional RRT algorithms for efficient re-planning in dynamic environments [C] //2007 IEEE International Conference on Automation Science and Engineering. IEEE, 2007: 1131–1136.
44. SALZMAN O, HALPERIN D. Asymptotically near-optimal RRT for fast, high-quality motion planning [J]. IEEE Transactions on Robotics, 2016, 32(3): 473–483.
45. KARAMAN S, WALTER M R, PEREZ A, et al. Anytime motion planning using the RRT [C] //2011 IEEE international conference on robotics and automation. IEEE, 2011: 1478–1483.
46. JANSON L, SCHMERLING E, CLARK A, et al. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions [J]. The International journal of robotics research, 2015, 34(7): 883–921.
47. BOOR V, OVERMARS M H, VAN DER STAPPEN A F. The Gaussian sampling strategy for probabilistic roadmap planners [C] //Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat.No.99CH36288C). IEEE, 1999, 2: 1018–1023.
48. TODOROV E, EREZ T, TASSA Y. Mujoco: A physics engine for model-based control [C] //2012 IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2012: 5026–5033.
49. KIM M C, SONG J B. Informed RRT* with improved converging rate by adopting wrapping procedure [J]. Intelligent Service Robotics, 2018, 11: 53–60.
50. GAMMELL J D, SRINIVASA S S, BARFOOT T D. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs [C] //2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015: 3067–3074.
51. LENZ I, LEE H, SAXENA A. Deep learning for detecting robotic grasps [J]. The International Journal of Robotics Research, 2015, 34(4–5): 705–724.
52. REDMON J, ANGELOVA A. Real-time grasp detection using convolutional neural networks [C] //2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015: 1316–1322.

53. VAHRENKAMP N, WIELAND S, AZAD P, et al. Visual servoing for humanoid grasping and manipulation tasks [C] //Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots. IEEE, 2008: 406–412.
54. LEVINE S, PASTOR P, KRIZHEVSKY A, et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection [J]. The International Journal of Robotics Research, 2018, 37(4–5): 421–436.
55. LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation [C] //Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 3431–3440

Chapter 6

3D Scene Understanding for Robot Interaction Tasks in Practice



6.1 Background

The understanding of specific scenes in the objective world is based on the analysis of the scene, explaining the content of the scene (explaining what objects are in the scene, where they are located, what relationships exist between objects, etc.), in order to achieve effective analysis of the scene. In recent years, scholars both domestically and internationally have paid close attention to this challenging research hotspot, and significant progress has been made in the research on robot scene understanding technology.

In the 1960s, the Shakey robot developed by Nilsson [1] at the Stanford Research Institute's Artificial Intelligence Center laid the foundation for future robot vision research by transmitting image data wirelessly to the main computer, which then returned wireless signals to control the robot's movement.

In the mid-to-late 1980s, vision research began to be combined with mobile robots, incorporating technologies like multi-sensor fusion technology, spatial geometric methods, and physical knowledge to achieve road tracking. Martin Marietta Company used vision and range sensors to build boundaries of roads and obstacles and successfully demonstrated road tracking in 1986 [2, 3], pioneering the early use of visual information to enable autonomous driving of robots on unstructured roads. Carnegie Mellon University developed the Navigation Laboratory (Navlab) [4, 5], which stored libraries of road and non-road color models, employed color classification models and neural networks to learn and identify road features, and developed three road recognition systems: Supervised Classification Applied to Road Following (SCARF), Yet Another Road Follower (YARF), and Autonomous Land Vehicle in a Neural Network (ALVINN). These three systems demonstrated good navigation performance in practical applications.

Due to the limitations of data collection technology, previous research was usually based on images, binocular vision, and other related technologies. Traditional scene labeling is mostly based on 2D color images. Since depth information cannot

be obtained, it is difficult to use it to achieve the reconstruction and understanding of 3D scenes. With the rapid development of 3D data collection technology, it is now possible to quickly construct a high-quality 3D scene model using methods such as ordinary three-channel color images + depth maps (RGB-D) sensors.

In recent years, the diversification of 3D data collection methods and the large amount of production software have resulted in an explosive growth of 3D data, making big-data-based 3D scene understanding possible. By combining data learning, biological understanding, and statistical modeling methods, it provides new solutions for the analysis and labeling of 3D object shapes.

The primary research focuses of 3D scene labeling technology is on automatic labeling techniques, as manual labeling is less efficient and therefore rarely used. In recent years, researchers have gradually incorporated advanced machine learning techniques into 3D scene labeling studies, with model segmentation and labeling based on supervised learning being the most common application. Kalogerakis et al. [6] proposed a data-driven approach for model segmentation and labeling, using a conditional random field (CRF) model to establish a learning objective function. This method employs hundreds of annotation features derived from geometric and contextual information and uses advanced Joint Boost techniques to train classifiers for mesh model triangle elements. Sidi et al. [7] adopted a similar approach for segmentation and labeling, combining content-driven analysis with knowledge-driven learning to achieve favorable segmentation results. The content-driven analysis here mainly refers to analyzing geometric similarity between shapes. Sunkel et al. [8] investigated the recognition of linear features in large-scale scanned point cloud data based on supervised learning, where users interactively outline several line features on the input point cloud as training data. The method then automatically learns and identifies the remaining features in the input data. Xu et al. [9] proposed a focus point analysis method, representing the 3D scene model as a graph structure. A focus point refers to a subgraph structure that frequently appears among a large number of related scene graph structures. By automatically extracting these context-related subgraph structures, it is possible to perform structural analysis and structured organization of large-scale scene datasets. Lai et al. [10] proposed the HMP3D (Hierarchical Matching Pursuit-3D) algorithm, which automatically extracts features from RGB-D images and point cloud data without manual design, enabling the identification of objects in small desktop scenes through learning.

At present, the existing 3D model databases [11–15] are mainly divided into two types:

One approach is viewpoint-centric method, where this labeling technique relies solely on a large number of two-dimensional images for object annotation. Since RGB-D images capture only a limited range of viewpoints of objects in the real world, only partial information can be obtained, as exemplified by the depth dataset from New York University [14], which includes scanning information of numerous scenes. Nonetheless, it remains a viewpoint-based dataset. Questions such as “What does it look like from behind this object?” and “What does the entire space look like from a unique viewpoint in reality?” cannot be answered.

Another approach is location-centric method, which annotates 3D scene models. The annotation results are spatial, and the necessary information can be obtained from various perspectives. However, most of the existing methods are based on the learning results of human object models. This method has high requirements for the reconstructed scene model, and because the shape of the same type of objects in the real world varies greatly, there is a certain bias in practical applications. There is also a method that uses two-dimensional images to annotate 3D models. This method is robust, but it still requires information such as the angle and position of the scanning camera, and it is very difficult to accurately estimate them in a large range of scene space.

Despite this, more and more experts and scholars are devoted to the technical reform of 3D scene understanding, gradually improving the size of data, scene complexity, quality of scene models, and computational process.

6.2 Overview

Humans can quickly recognize objects in a scene, but due to the inherent complexity of visual problems, the richness of scene understanding information, and the large computational load, it is difficult for robots to match human understanding capabilities. As a crucial part of robot scene understanding, the construction and comprehension of 3D scenes are key sources for robots to perceive their surroundings. The ability to accurately and in real-time process visual information directly affects the behavior of robots, which is especially critical in environments characterized by randomness and complexity. Therefore, 3D scene understanding technology is a critical aspect of research in robotics. Here, the term “3D scene understanding” refers to the robust perception of the environment from local to global scales, from image appearance to geometric shape, and the abstraction from low-level data to meaningful entities, along with the inference of environmental state through the interpretation of these entities. This information aids robots in completing tasks such as control and movement. 3D scene understanding is an advanced visual technology that plays a significant role in studying robotic comprehension capabilities. However, due to its high demands on data volume, scene complexity, and computational processes, it has not been extensively researched or applied in past robotic vision systems.

The purpose of this project is to expand the application of robot vision understanding calculation methods by addressing potentially incomplete reconstructed scenes and establishing a primary framework for robots to achieve scene visual understanding. Through the practise of this project, robots will be able to comprehend 3D scenes more accurately, ultimately enabling them to complete movement and recognition tasks based on their level of visual understanding of the environment.

6.3 Framework

The 3D scene understanding system based on robotic technology is essentially a conceptual framework. It not only involves the control of robots and the use of platforms but also encompasses the transformation and processing from 2D to 3D in the field of computer vision. According to the project's operational workflow, the 3D scene understanding system based on robotic technology can be divided into four main components:

1. **Driving and Information Collection of Multiple RGB-D Cameras.** To achieve the integration of multiple cameras and perspectives, the first challenge is to synchronize and collect information from multiple RGB-D cameras within the same environment.
2. **3D Reconstruction of Indoor Scenes Based on Multi-RGB-D Camera Fusion.** 3D reconstruction is the foundation of the entire scene understanding system and serves as the basis for robot control and 3D model creation.
3. **Scene Annotation Techniques Based on RGB-D Sequences.** Key frames are selected from the captured RGB-D sequences and manually annotated. These frames are then transmitted through the cloud, with the annotation results mapped onto the 3D scene model.
4. **Simulation Platform Design.** The simulation platform includes modules for information collection, 3D scene reconstruction, scene cloud annotation, and settings. The information collection module features a submodule for real-time display and storage of the robot's real-time path.

6.4 Module Design and Implementation

6.4.1 *Driving and Information Collection of Multiple RGB-D Cameras*

6.4.1.1 Equipment and Environment

Figure 6.1 shows the conceptual diagram of the panoramic view indoor robot platform designed by the project team, and Fig. 6.2 is an actual photo of the platform. The platform uses a TurtleBot2 robot equipped with an open-source Robot Operating System (ROS) and is fitted with an ASUS mini host. The mini host connects to the TurtleBot2 robot and is responsible for robot control and driving multiple RGB-D cameras. The mini host runs the Ubuntu system and is configured with the necessary environment. Four Xtion Pro Live cameras are mounted on top of the robot. These cameras form an array at different angles and directions and are connected to the mini host via USB ports. After installing the camera drivers, the system can complete the information collection and processing tasks of the indoor

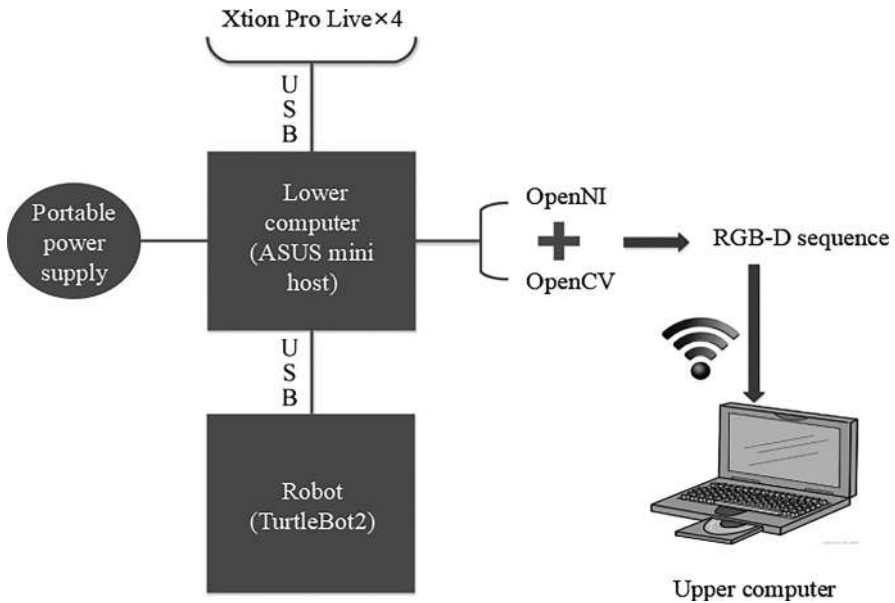


Fig. 6.1 Concept diagram of the panoramic view indoor robot platform

scene. In this section, this mini host is referred to as the “slave computer,” which is connected to the computing platform via a wireless network. The computing platform, referred to as the “master computer,” is responsible for receiving the RGB-D sequences transmitted by the lower computer and completing the 3D scene reconstruction from these sequences.

TurtleBot2 and ROS

TurtleBot is a cheap, open-source software-equipped personal robot manufactured by Willow Garage in 2010. Users can conveniently download the TurtleBot Software Development Kit from the ROS wiki website for development. As an entry-level mobile robot platform, TurtleBot is easy to purchase, manufacture, and install, and it is easy to combine with other devices. The TurtleBot platform can be used to implement real-time obstacle avoidance, automatic navigation, real-time positioning, and map building (SLAM) algorithms and can be controlled via a laptop or smartphone.

This project selects the second-generation TurtleBot robot as the platform for scene scanning. The adjustable frame of the TurtleBot2 robot (see Fig. 6.3) allows for convenient placement of the mini host and cameras, while the chassis’ movement and rotation functions meet the robot’s indoor mobility requirements.

The TurtleBot2 robot is equipped with ROS, a well-known open-source operating system in the robotics field. Developed by Willow Garage, ROS is designed

Fig. 6.2 Photo of the panoramic view indoor robot platform



specifically for robot software development. It includes a vast array of software libraries and tools that assist users in building their own robot applications. With its extensive drivers and powerful development tools, ROS has significantly simplified the development and testing of robotic programs.

This project uses Ubuntu 14.04 installed on the lower computer (i.e., the ASUS mini host carried on the TurtleBot2 robot bracket) and controls ROS in Ubuntu 14.04 to implement the robot movement operation program. Ubuntu is a Linux operating system installed on personal computers and is a free and open-source operating system. This system can conveniently drive the camera and do application development through environment configuration.

Fig. 6.3 TurtleBot2 robot**Fig. 6.4** Popular RGB-D cameras. (a) Kinect 1.0; (b) Kinect 2.0; (c) Xtion Pro Live

Xtion Pro Live and Camera Array

The popular RGB-D cameras include the Microsoft Kinect 1.0, Microsoft Kinect 2.0, and the ASUS Xtion Pro Live (see Fig. 6.4). Compared to the Microsoft Kinect cameras, the ASUS Xtion Pro Live is smaller and more portable. It does not require a power adapter, as it can be powered directly through a USB 2.0 interface. Additionally, it uses the Open Natural Interaction (OpenNI) driver, which allows it to operate in a cross-platform OpenNI environment. Therefore, the ASUS Xtion Pro Live is used in this project's robot platform to collect depth and color information of indoor scenes.

1. Introduction to Xtion Pro Live Sensor: The Xtion Pro Live sensor is a low-cost depth sensor released by the ASUS Group, which uses the 3D data acquisition technology of the Israeli company PrimeSense. Compared with Microsoft's similar depth sensor Kinect, the Xtion sensor has the advantages of smaller size, lighter weight, and more convenient use. Using the Kinect sensor requires

connecting an interface converter and an external power supply, which is very inconvenient.

The Xtion depth sensor uses an infrared camera and depth adjustment detection technology to capture the depth data of target objects. Originally developed to capture real-time human limb movements and support accurate tracking of dynamic body movements, it was intended for applications like motion-sensing games. However, due to its advantage in quickly acquiring depth information, it has increasingly been used for environmental data collection. The development solution for Xtion provides users with a set of tools, enabling the creation of gesture recognition platforms without needing to write low-level code. Currently, the Xtion sensor system offers two main functionalities: gesture detection and full-body tracking. For gesture detection, the Xtion sensor accurately captures hand movements with no delay, allowing users to control games using their hands instead of a joystick. It can detect various actions such as clicks and rotations, and can be integrated into application control interfaces, facilitating the development of multiple applications. The full-body tracking functionality enables the sensor to track overall body movement, which is useful for projects involving full-body design. Consequently, the Xtion sensor can be employed in a variety of interactive applications, including motion-sensing games, medical applications, and online meetings. Additionally, the Xtion development kit is completely open-source, making it highly convenient for developers.

In order to convert abstract information into data information, the Xtion sensor needs components for laser output and receiving input, as shown in Fig. 6.5. The Xtion sensor can be divided into several parts: the infrared laser transmitter and receiver part, the chip part for data processing, and even the sound and color processing parts needed for a complete function combination, such as the microphone and RGB color camera components, etc. Because it is necessary to convert abstract information and actual data information, the Xtion sensor needs to

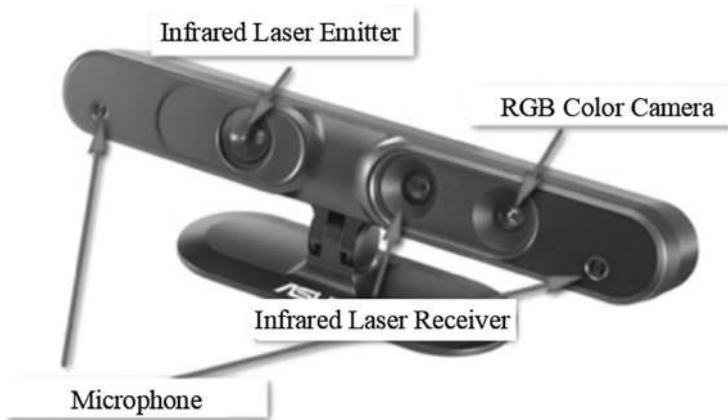


Fig. 6.5 Xtion structure diagram

process the spatial abstract information obtained through the infrared transmitter through a series of chip processing, so as to obtain the depth data of the image, which is actually similar to a sensor.

The wide application of the Xtion sensor is also due to its ability to incorporate diversified information integrated processing, such as the aforementioned sound and color, and it can also add voice information, etc., to make it more interesting and realistic in somatosensory games and other new human-computer interaction. Some inherent parameters of the Xtion sensor: horizontal viewing angle of 57° and vertical viewing angle of 43° . Output parameters include depth bit and sensitivity. The frequency of the RGB image output and the converted depth map is 30 Hz, the image resolution is generally 640×480 specifications, and the real distance represented by the depth map obtained is 80–350 cm.

2. Working Principle and Architecture Support of Xtion Pro Live: The Xtion sensor needs to convert abstract infrared images into depth images, which involves using specific encoding techniques to transform abstract features into machine-processable information. The Xtion sensor employed light encoding technology. Specifically, this involves emitting laser speckles from a light source through an emitter onto objects in 3D space, which generates random diffraction speckles on the object's surface. These randomly generated diffraction speckles are unique to different surfaces and positions on the object. It is this uniqueness that allows the differentiation of characteristics at various positions of the same object. In other words, the uniqueness of these feature points enables the recognition of the 3D information of a specific object. The core processing chip of the device is tasked with decoding the previously acquired 3D information to produce the final depth map. In summary, the Xtion sensor utilizes light encoding and decoding processes to convert real-world images into depth images that can be recognized by machines.

The front of the ASUS Xtion Pro Live camera features three lenses: the far left is the infrared emitter, the far right is the infrared receiver, which together are used to sense depth information, and the middle lens is the RGB camera, used for capturing color images. The Xtion Pro Live camera has an effective range of 0.8–3.5 m, with a depth camera accuracy of 3 mm at approximately 1 m. The accuracy decreases significantly with increased distance, being about 3 cm at 3 m. The depth camera has effective viewing angles of 58° horizontal, 45° vertical, and 70° diagonal. These viewing distance and angle parameters imply that when a single camera scans an entire indoor scene, the camera's direction, and pitch angle must be adjusted at the same position, which significantly reduces information collection efficiency, increases the time and distance required for the robot's operation and movement, and raises the cost of indoor scanning.

Based on the above considerations, multiple ASUS Xtion Pro Live cameras are used to form an RGB-D camera array, thereby "enlarging" the robot's field of view. As shown in Fig. 6.6, since the indoor height is generally 3–4 m, in order to obtain the 3D information of the entire indoor scene, considering the effective viewing distance of the Xtion Pro Live sensor of 0.8–3.5 m, the plat-

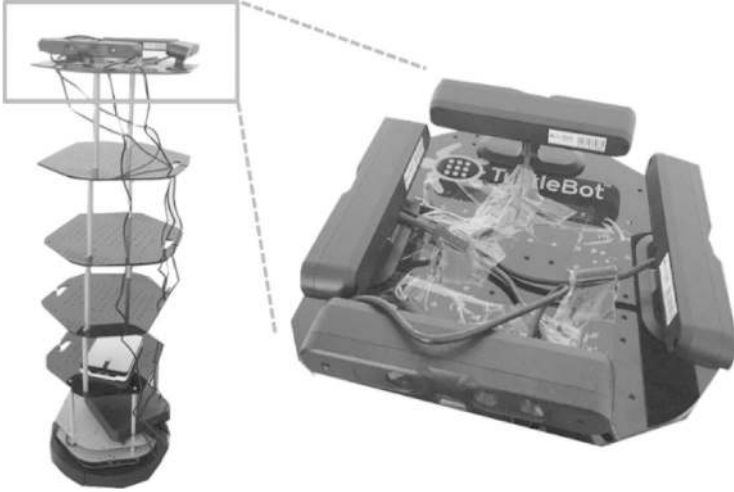


Fig. 6.6 RGB-D camera array

form raises the TurtleBot2 robot to 1.3 m through a bracket, and places the camera on the flat top of the robot, so that the scanning distance meets the height of the indoor scene.

In order to obtain a wider field of view, the platform places four ASUS Xtion Pro Live cameras on the top of the robot to form an RGB-D camera array. The four cameras are facing different angles, the diagonals of the camera center are perpendicular to each other, and the camera tilt of the Xtion Pro Live camera can be adjusted without segmentation. By adjusting the tilt angle up and down, information from different viewing areas in the vertical direction can be obtained. The platform sets the tilt angles of the four ASUS cameras to 5° up, 10° down, 25° down, and 40° down. The reason for this design is that in the experiment, it was found that the field of view of the pitch angle is mostly simple objects such as walls and shelves, while the field of view obtained by looking down is mostly cluttered objects. Therefore, there needs to be more overlapping areas in the field of view where the camera is looking down to obtain more accurate scene reconstruction results. With such an arrangement of tilt angles, a 30° overlap area can be obtained in the field of view of two adjacent tilt angle cameras, which will be beneficial to the implementation of the loop closure process during scene construction.

The camera array, consisting of four ASUS Xtion Pro Live cameras with varying tilt angles and orientations, allows the robot to obtain a panoramic view of the position by rotating once. Additionally, there is a 30° overlap between the fields of view of adjacent cameras, which significantly enhances both the efficiency of information collection and the accuracy of scene reconstruction results.

6.4.1.2 Robot Panoramic View Scanning

Scanning Path

In order to obtain a broader view and make the reconstruction results more accurate, the robot needs to rotate with a camera array to obtain a panoramic view and the overlapping area between adjacent views. The system first needs to define the scanning area as a sub-scene, which divides the indoor space into multiple sub-scenes. The robot's scanning path is designed so that it stops and rotates twice in place for every 30 cm of forward movement, and then continues moving forward. By sending instructions from the upper computer to the lower computer, the lower computer can control the robot to follow this simple path for walking and information collection, thus completing the scanning of each indoor sub-scene. As for controlling the robot's movement to each sub-scene, the system of this project still requires manual operation commands to be implemented.

Manual Control

The software has implemented remote manual control of robot movement, specifically including rotation instructions and straight walking instructions. The upper computer serves as a socket server, sending string instructions to the lower computer robot computer socket client. The lower computer receives the string and analyzes whether it is a rotation instruction or a straight walking instruction, driving the mobile platform to perform the corresponding movement. After each command sent by the upper computer, the robot rotates 10° or walks straight 10 cm.

Automatic Movement

The robot moves automatically along a predetermined path during the initial collection of scene depth maps. The lower computer has planned the robot's path, with the robot rotating 720° and advancing 0.3 m each time, which can achieve a good scene scanning effect.

Fixed-Point Movement

After reconstructing the 3D scene map, the robot will move to the position of a target object based on the coordinates. The robot's coordinate system is a planar coordinate system, with the robot's center as the origin. The positive x -axis points forward from the robot, and the y -axis points to the left. This means that you only need to provide the target object's coordinates (x, y) to drive the robot to the specified point. To minimize the travel distance, the strategy is to first rotate the robot to align the positive x -axis with the direction of the target object and then move in a

straight line. The rotation angle is calculated using trigonometric functions, and the target distance is calculated using the Pythagorean theorem. The distance measured by the odometer is compared with the target distance to determine if the robot has reached the specified position. If it has, the movement is stopped.

Obstacle Avoidance

In order to improve the robot's adaptability to the environment, a simple obstacle avoidance function that does not rely on the map has been added, mainly including depth camera ranging and path planning.

Depth Camera Ranging

The real-world environment is very complex, consider a simple situation where the robot avoids obstacles without a map. First, set a dangerous distance value $D_{\min}$. After passing the coordinates of the target object to the robot, the robot continuously obtains the distance value D of the nearest object during its movement. Before reaching the target object, if the robot encounters an obstacle (i.e., $D \leq D_{\min}$), it rotates to find other paths until $D > D_{\min}$, recalculates the rotation angle and distance with the target object, and continues to move forward in the manner of fixed-point movement. If it encounters an obstacle, it uses the same method to avoid it.

Implementation of Synchronous Acquisition with Multiple Cameras

1. Listing of environmental equipment: Due to the need to connect multiple RGB-D cameras to the same host, the creation of the device environment involves having multiple device nodes in the same environment. It is essential to enumerate and number the devices for efficient management. The process starts by listing all nodes under the specified conditions and placing the results into a list. These results are then established as nodes in the environment to facilitate data stream generation. Finally, device nodes are created based on the information in the list. After creating the depth device node and RGB device node according to the corresponding conditions, the appropriate generator is obtained. By setting the mode of the generator, the enumeration of all device nodes and the creation of the depth and RGB generators can be completed. The platform used in this project acquires depth and RGB information with a resolution of 640×480 and a frame rate of 30 fps during robot collection.
2. Data update: After creating all the depth generators and RGB generators in the environment, put them into two containers respectively, and start all generators in the environment through the API interface, that is, start the data stream. Then wait for the data to be updated, use the generator in the container to obtain the

metadata, and the validity of the frame can be judged by comparing the current frame number with the previous frame number.

3. Data format: To effectively complete the upcoming scene construction algorithm, a unified and manageable data format is required. The RGB images used in this project are in Mat format, which can be processed by the open-source computer vision library (OpenCV). These images are stored on the master computer in Joint Photographic Experts Group (JPG) format, with a resolution of 640×480 , derived from metadata conversion according to the generator mode. The depth maps are in 16-bit Portable Network Graphics (PNG) format. To facilitate easier viewing, the depth map is shifted by 3 bits. Similarly, during the preprocessing stage of the scene construction algorithm, a reverse shift of 3 bits is necessary. Additionally, during data collection, each frame's frame number and timestamp are recorded to aid in processing by the scene construction algorithm.

6.4.1.3 Transmission and Processing of RGB-D Sequences

Through the creation of multi-camera device nodes and generators, continuous frame depth map sequences and RGB image sequences from different perspectives can be obtained during the robot's movement. However, the RGB-D sequences from multiple cameras with varying fields of view need to be transmitted efficiently to the master computer. The host computer then processes and saves the received sequences locally for offline scene construction algorithms.

Information transmission: In this project, the platform used transmits information over a wireless connection, with the slave computer and the master computer communicating via Transmission Control Protocol/Internet Protocol (TCP/IP). For each data update, each generator in the generator container sequentially retrieves metadata and processes images. In an environment with four RGB-D cameras, each data update requires transmitting eight frames of data, amounting to nearly 50 MB of data per second at a frame rate of 30 fps. To ensure stable transmission, the image data is compressed before transmission and decompressed at the master computer. For each frame of data, the transmission protocol is: first send the image size, then the image's save path and filename for setting the storage path on the master computer, and finally send the image data.

RGB-D sequence processing: The save paths sent by the slave computer are organized into different folders according to the respective generators. The filename is crucial for processing the RGB-D information. Both depth maps and RGB images follow the naming convention "frame number-timestamp." These identifiers are obtained from the generator when metadata is retrieved on the slave computer. The master computer decompresses the received images according to this naming rule and saves them in the corresponding paths. These images are then used as inputs for the subsequent scene construction algorithm.

6.4.1.4 Scanning and Collection Results

This project places four ASUS Xtion Pro Live cameras on top of the TurtleBot2 robot and connects them to the Asus mini host ROG GR8 at the bottom. The mini host, equipped with an i7 processor and 16 GB of memory, is fully capable of collecting and transmitting RGB-D information. It is important to note that since the host needs to connect to the robot and four RGB-D cameras, at least five USB ports are required. Additionally, in the OpenNI environment, each USB port can only recognize one device, so using USB hubs or similar tools is ineffective, which limits the choice of standard slave computer. On this platform, the five RGB-D cameras and the robot all need to be powered through the mini host's USB ports. Therefore, the project equips the mini host (i.e., the lower-level machine) with a portable power supply to facilitate scanning work with the robot (the robot platform can be seen in Fig. 6.3).

As mentioned earlier, the platform built for this project configures the four ASUS cameras with tilt angles of 5° upward tilt, 10° downward tilt, 25° downward tilt, and 40° downward tilt. During laboratory experiments, the research room was divided into three straight corridors for scanning, and the collected RGB-D images were transmitted to the host computer.

In the master computer, the received images are decompressed and saved locally using OpenCV. Figure 6.7 shows the first frame of depth and color images obtained synchronously by the four RGB-D cameras in the research room scene.

From Fig. 6.7, you can see two different camera views. In the same view, the RGB image provides color information, and the depth map provides distance information relative to the camera plane.

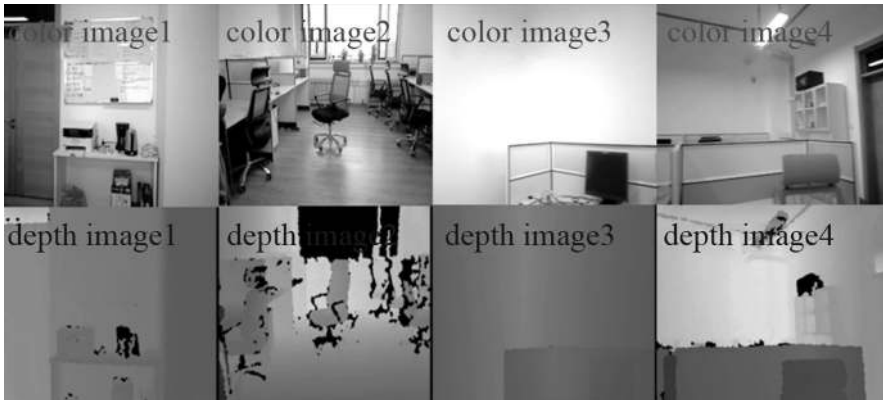


Fig. 6.7 Multiple RGB-D cameras synchronously display

6.4.2 3D Reconstruction of Indoor Scenes Based on Multi-RGB-D Camera Fusion

With the entry of consumer-grade depth cameras into the public eye, 3D scene reconstruction has become a popular research topic. Existing methods still face challenges such as low information collection efficiency, memory limitations on scene construction, and difficulty in direct application to robots, especially for large-scale indoor scenes. This project proposes a method for 3D reconstruction of indoor scenes based on Multi-RGB-D camera fusion. For extensive and generalized indoor scenes, the method first improves information collection efficiency by fusing the views from multiple RGB-D cameras. Next, it applies a data-driven algorithm, specifically the Structure from Motion (SfM) algorithm, to reconstruct the scene based on the collected RGB-D sequence frames. Finally, it uses a point cloud stitching algorithm based on the nearest point iteration to achieve a 3D representation centered on the scene.

The purpose of this project practice is to achieve large-scale indoor scene reconstruction. Through online multi-camera synchronous scanning, offline scene construction, and stitching, the indoor scene reconstruction is successfully implemented on the robot TurtleBot2, and the information collection efficiency and scene construction results of the system are verified in terms of range superiority.

6.4.2.1 Research Content

The research content of this project practice is 3D reconstruction of indoor scenes based on Multi-RGB-D camera fusion. For large-scale, generalized indoor scenes, from the driving and information collection of Multi-RGB-D cameras, scene construction based on RGB-D sequence frames, and calibration and fusion of multiple camera perspectives, with the scene-centered scene representation as the basis, the purpose is to achieve large-scale indoor scene reconstruction and complete the robot's perception and modeling of the indoor scene.

Driving and Information Collection of Multiple RGB-D Cameras

Information collection is a prerequisite for indoor scene reconstruction. To enable the robot to achieve a panoramic view during the information collection process and to enhance the speed and efficiency of data acquisition, this project investigates how to perform synchronous scanning with multiple RGB-D cameras oriented toward the robot, and how to format and output the RGB-D frames collected by the cameras, thereby completing the RGB-D information collection for the entire indoor scene. The research focuses on two main aspects:

- First, it explores how to establish the robot's scanning path within the indoor scene to ensure comprehensive scene coverage.
- Second, it examines how to operate multiple RGB-D cameras and process the collected visual information. The research content has been detailed in the section "Driving and Information Collection of Multiple RGB-D Cameras."

Scene Construction Based on RGB-D Sequence Frames

Scene construction is key to indoor scene reconstruction. To reconstruct the scene from the images collected by RGB-D cameras and achieve a scene-centered representation, this section implements a data-driven structure from motion (SFM) algorithm. For the RGB-D sequences collected by multiple cameras, the pose of each camera is calculated to complete the construction of multiple scenes. This includes the following two aspects:

- First, research on how to estimate the camera pose by detecting the matrix transformation relationship between adjacent frames in the collected RGB-D sequence.
- Second, research on how to construct a 3D scene based on the RGB-D sequence and the estimated camera positions using a data-driven method.

Multi-Camera Perspective Calibration and Fusion

The fusion of multiple perspectives is an effective method to improve the range and accuracy of indoor scene reconstruction. To obtain a 3D model of the full-range indoor scene and achieve reasonable fusion of multiple sub-scenes, this project has researched how to calibrate the pose of multiple RGB-D cameras. For the RGB-D sequences obtained from multiple camera views, the initial pose of the cameras is corrected through bundle adjustment, aligning multiple camera views and achieving the fusion of multiple scenes. This includes the following two aspects:

- Firstly, research on how to calibrate multiple camera views and initialize the calibration of internal and external parameters of multiple cameras, as well as the relative position and angle information between cameras. Bundle adjustment is used to achieve global optimization of the corresponding camera poses for the RGB-D frames, improving the accuracy of scene reconstruction.
- Secondly, research on how to align and stitch sub-scenes, and achieve a 3D representation of large-scale complex indoor scenes through sub-scene stitching.

In summary, the research goal of this project is to enable robots to connect multiple RGB-D cameras and synchronize the camera views. Through synchronized multi-view collection, visual information of indoor scenes can be obtained more quickly. Based on the RGB-D sequences collected by multiple cameras, the

sub-scenes in each camera view are fused and further corrected to improve accuracy, resulting in a 3D representation of the entire indoor scene. Ultimately, the robot can reconstruct indoor scenes of any range with simple path planning, meeting the needs of indoor service robots for scanning and 3D map construction. Therefore, this project has practical significance and application value.

6.4.2.2 Scene Construction Method Based on RGB-D Sequence Frames

For the large number of 2D images collected by the camera, a robust algorithm is required to construct a 3D point cloud. Since the KinectFusion algorithm only uses depth information, it cannot construct color images obtained by the RGB-D camera in real time through point cloud fusion, and such an algorithm is not suitable for the vast amount of information in indoor scenes. Therefore, at this stage, an online scanning and offline computing mode is more appropriate for the construction of indoor scenes by robots.

The traditional SFM algorithm is used to calculate the 3D information from a large number of offline images. This algorithm can predict the 3D scene structure and camera positions from 2D images. Since each frame of images obtained by the robot during indoor scanning is continuous, and the depth information can assist in calculating the point positions in the RGB images, this RGB sequence can make the SFM algorithm more accurate and faster. The data-driven SFM algorithm based on RGB-D sequences can construct the corresponding 3D scene model from the continuous RGB-D information collected by the camera.

This method calculates the rigid transformation between cameras in images by detecting overlapping areas between adjacent frames. It then obtains the initial pose of the camera corresponding to each RGB-D frame by successively applying the rigid transformations between frames. Finally, the scene model is constructed by transforming the depth information into the camera coordinate system. Since the depth map corresponds one-to-one with the RGB image during the information collection process, the algorithm generates a 3D model with RGB information and saves it in the polygon file format (PLY). The research route and goals of the algorithm are shown in Fig. 6.8.

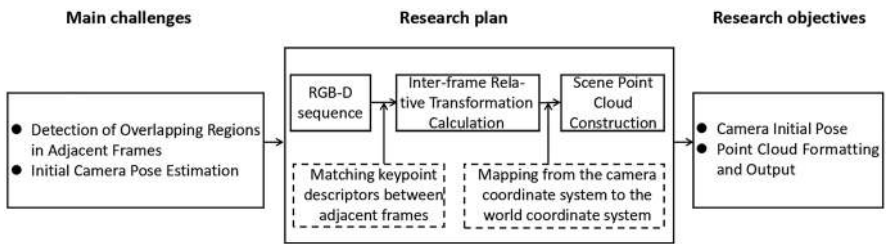


Fig. 6.8 Research route for scene construction based on RGB-D sequence

Two Representations of the Scene

In human visual understanding, there are two ways to represent the scene and spatial layout of the visual information obtained. One is based on the field of view, and the other is based on the center of the scene. This has been proved in the research of neuroscience [16], and there are corresponding areas in the human brain to do scene memory, navigation, route learning, etc. for this layout. The vision of robots is actually the same. Robots can obtain the existing targets and their relationships in the current field of view based on the information in the field of view, or they can center on the scene, establish the spatial layout in the scene, and master all the targets and their positions in the entire scene. When the robot wants to reach a certain target, it can reach the target through path planning and navigation. The scene representation used in this project is centered on the scene, collecting all the information of the entire scene and establishing a 3D point cloud of the scene. Through this representation, all objects in space are in the same model, laying the foundation for the next scene understanding.

Fast Storage and Organization of 3D Information

During the 3D scene modeling process, it's crucial to efficiently organize the collected 3D data within the storage system to ensure both speed and accuracy in the construction of geometric surfaces. Traditional methods use 3D arrays to represent scene space. However, this approach can lead to memory waste when scene objects are sparse and lack the flexibility to adapt dynamically when the scene is large, thus limiting the scale of the scene that the robot can manage.

Based on this, an octree structure constructed using GPU parallel processing is proposed. The octree structure allows for more flexible memory allocation, effectively avoiding memory waste and reducing the algorithm's memory requirements, enabling the robot to reconstruct larger-scale scenes. Additionally, the octree structure is well-suited for GPU parallel processing, ensuring that the algorithm can quickly handle scene information. As illustrated in Fig. 6.9, the upper half shows a schematic diagram of the KinectFusion algorithm, while the lower half depicts a 3D reconstruction using the octree. The core part of the proposed algorithm is highlighted within the red dashed box.

The octree structure is divided into four layers: the top layer, branch layer, middle layer, and data layer. The top layer is utilized to accelerate ray casting (ray casting is employed to predict the environment point cloud observed by the current depth sensor based on the existing model and current camera pose, which is then provided to the system for matching and fusion). Nodes in the branch layer serve as the root nodes for their corresponding subtrees. The middle layer stores all nodes relative to the branch and middle layers, while also maintaining an index pointing to its first child node, which is used for breadth-first traversal of the octree. The data layer stores other relevant information, such as the distance from the node center to the scene.

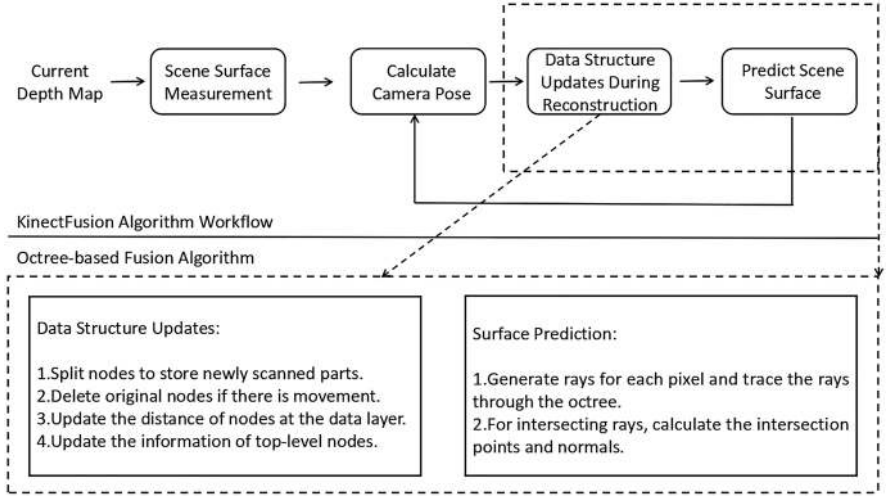
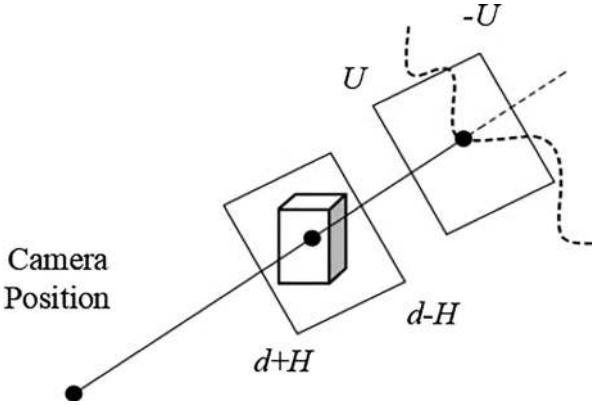


Fig. 6.9 KinectFusion algorithm and data management algorithm based on octree

Fig. 6.10 Node splitting judgment process



The update process of the octree during scene construction is as follows:

1. **Splitting nodes.** By traversing the octree in breadth-first order through the branch layer, and adaptively splitting nodes as needed. Parallel prediction can be chosen to determine whether the node needs further splitting. The specific judgment process is as follows: first, determine whether the node is in the view frustum, and calculate the distance from the node center to the current depth map. Then, determine whether the node contains scene information and whether it needs further splitting. The judgment principle is shown in Fig. 6.10.

Here, d represents the distance between the node center and the expected surface, and H represents the radius of the node's circumscribed sphere. Therefore, the distance from any point within the node to the expected surface falls within the interval $[d - H, d + H]$. U represents the maximum value of d ,

meaning that nodes outside the interval $[-U, U]$ are not considered and do not require splitting. Consequently, nodes that have an intersection between the intervals $[d - H, d + H]$ and $[-U, U]$ need further splitting. The method of parallel prefix scan primitives, as described in reference [17], is employed to compute and store child nodes while maintaining a variable that tracks the number of split nodes.

2. Deleting nodes needs to consider that there may be dynamic moving objects in the scene, and the nodes where the objects were originally located need to be deleted after movement to save memory. Node deletion can be seen as the reverse process of node splitting, which can be traversed from the data layer to the branch layer from bottom to top.
3. SDF update. After the octree structure is updated, the fusion algorithm in reference [18] is used to update the distance information in the data layer. According to the coordinates of the leaves, calculate the distance from the center of the leaf node to the scene and update the weight.
4. Finally, after the octree construction of the scene geometry data is completed, the scene surface under the current viewpoint is calculated and constructed on the octree through the GPU-based ray tracing method, finally completing the rapid construction of the 3D geometry data scene.

SFM Driven by RGB-D Data

Traditional structure from motion (SFM) techniques rely on unordered RGB images to determine camera poses and use bundle adjustment for corrections. The 3D coordinates of RGB points are then calculated based on these poses using photogrammetry principles, followed by Poisson surface reconstruction from sparse point clouds. However, due to the absence of depth maps, this method yields lower accuracy when calculating the 3D positions of points. In contrast, using RGB-D sequences to compute camera poses across continuous frames offers greater precision. The depth map assists in calculating 3D coordinates, reducing computational complexity and enhancing the overall accuracy. This project specifically addresses the multiple RGB-D sequences acquired by the lower-level system, implementing this data-driven SFM. The following sections introduce the algorithm and its implementation process.

Motion recovery structure is a range imaging technique. Computer vision refers to a complex process of continuously analyzing the motion of an object to obtain complete 3D structural information of the object. In layman's terms, it is the process of collecting and reading 2D image sequences of real scenes and estimating the complete 3D structure of the corresponding scene through the corresponding matrix transformation algorithm.

Humans can perceive and acquire a great deal of 3D structural information about their environment through their movement. When either the observer or the observed object moves around the observer, 3D information is derived from the continuously

changing 2D image data over time. Finding structure from motion is fundamentally similar to finding structure from stereoscopic vision; both methods focus on determining the correspondence between points in 2D images and voxels in a 3D model.

In computer vision, a camera model is used to relate points in the 3D model to points in the 2D image. Most 3D reconstruction techniques currently employ the pinhole camera model as the foundational model. To fully understand this model, one must first grasp the concept of four coordinate systems [9]: the image pixel coordinate system, the image physical coordinate system, the camera coordinate system, and the world coordinate system.

1. Image pixel coordinate system: A plane rectangular coordinate system with pixels as units. The upper left corner of the image is the origin, and the coordinates (u, v) of each pixel correspond to the number of rows and columns in the image.
2. Image physical coordinate system: A plane rectangular coordinate system with millimeters as units. The origin is the intersection of the lens axis and the imaging plane. The x -axis is parallel to the u -axis of the image coordinate system, and the y -axis is parallel to the v -axis.
3. Camera coordinate system: A 3D coordinate system (X_c, Y_c, Z_c) with the camera's optical center as the origin O_c . The X_c and Y_c axes are parallel to the x -axis and y -axis of the image physical coordinate system, respectively, and the Z_c axis is the camera's optical axis, perpendicular to the image plane. The camera's focus center is the origin, forming the camera coordinate system.
4. World coordinate system: A reference coordinate system is chosen in space to represent the camera's spatial position and to describe the position of any object within the spatial environment. The world coordinate system is a 3D coordinate system that can be arbitrarily defined by the user and serves as the absolute coordinate system for the objective world. The origin, O_w , is defined with the X_w axis pointing horizontally to the right, the Y_w axis pointing vertically downward, and the Z_w axis determined by the right-hand rule. The relationship between the world coordinate system and the camera coordinate system can be represented by a rotation matrix and a translation vector. If the three axes of the world coordinate system are (X_w, Y_w, Z_w) , then

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (6.1)$$

where R and T represent the rotation matrix and translation vector of the camera, respectively. The imaging process of the camera needs to be completed through three transformations between these four coordinate systems. First, the information in the world coordinate system needs to be converted to the camera coordinate system; then the transformation from the camera coordinate system to the image physical coordinate system needs to be completed. Finally, the

information in the image physical coordinate system is converted to the image pixel coordinate system.

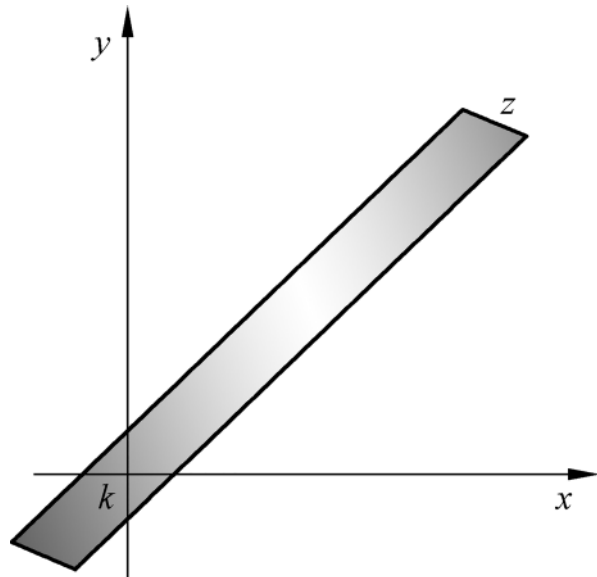
For the reconstruction process, there are traditional matrix reconstruction methods and sequential reconstruction methods. Each step of sequential reconstruction first needs to restore the motion of the camera, that is, the projection matrix, and then use triangulation to restore the structure of the scene, so the sequential reconstruction method is also called motion recovery structure.

1. Camera Pose Calculation: During the acquisition process, there is significant overlap between adjacent RGB-D frames, resulting in small distances between keypoint descriptors, which improves keypoint matching accuracy. Additionally, RGB-D images from the same viewpoint may appear multiple times. In the loop closure process, matching nearly identical RGB-D frame pairs helps effectively close the loop. This approach extends existing data-driven methods used in computer vision problems.

As shown in Fig. 6.11, the $x - y$ plane represents the plane where the depth sensor is located, point k represents the position of the sensor, that is, the coordinate origin. The center of the 2D image is projected onto the corresponding point on the plane where the sensor is located, and the depth value corresponding to each point on the depth map is its distance to the plane where the sensor is located, that is, the depth value is its z -axis coordinate. The direction pointed by the z -axis is also the direction pointed by the sensor lens.

To determine the camera parameters of the world coordinates where the sensor is located, it is first necessary to determine the initial viewing angle of the camera, that is, the horizontal and vertical viewing angles of the sensor. As shown in Fig. 6.12, angle α and angle β represent the angle values of the horizon-

Fig. 6.11 3D coordinates with the sensor as the origin



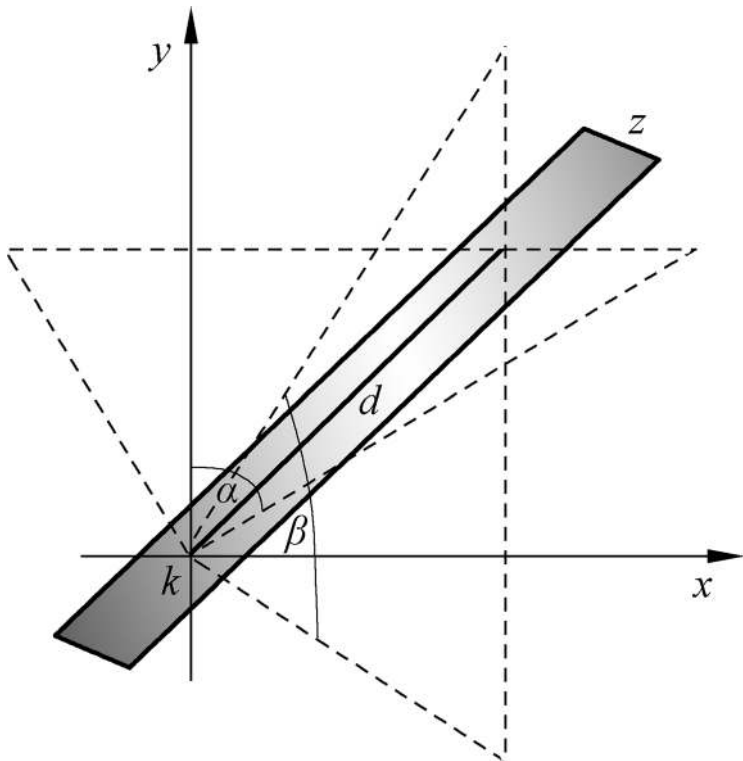


Fig. 6.12 Schematic diagram of sensor viewing angle

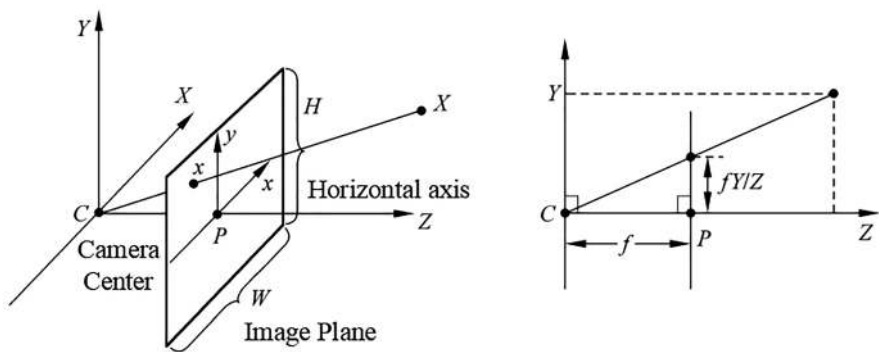


Fig. 6.13 Coordinate calculation analysis diagram

tal and vertical viewing angles of the sensor, respectively. After obtaining the values of angle α and angle β , according to Fig. 6.13, the ratio of the width value W of the 2D image to the distance d is $2 \tan (\alpha / 2)$, and similarly, the ratio of the height value H to the distance d is $2 \tan (\beta / 2)$. The format of the camera parameter matrix is

$$K = \begin{bmatrix} A & 0 & B \\ 0 & C & D \\ 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

In formula (6.2), A represents the ratio of the pixel width of the 2D image to $2 \tan(\alpha/2)$, B is half of the pixel width, C is the ratio of the pixel height to $2 \tan(\beta/2)$, and D is half of the pixel height. At this point, the camera parameter matrix can be correctly calculated.

- (a) Registration of adjacent frames: Since the RGB-D sequence collected is continuous, the overlap between adjacent frames is significant, allowing for alignment through feature point matching. The algorithm starts by using scale-invariant feature transform (SIFT) to match key points between consecutive frames. SIFT key points are effective for detecting and describing local features in images and are commonly used in object recognition, 3D modeling, and image stitching. Poor matches are then filtered out using a ratio test, and valid depth values are selected from the set of SIFT key points. Finally, the relative transformation between frame pairs is computed using the random sample consensus (RANSAC) algorithm, which is employed for estimating the camera pose.
- (b) Loop closure: To detect loop closure, a “bag of words” model is used to compute feature vectors for each frame. First, for each RGB-D sequence, the K-means algorithm trains the codebook for the “bag of words” model. For each frame in the sequence, scale-invariant feature transform (SIFT) is used to extract features, and a visual word histogram is created. In this histogram, the weight of each word is the inverse of its frequency in the RGB-D sequence, following the standard term frequency-inverse document frequency (TF-IDF) method. Next, the dot products of feature vectors for all frame pairs are computed to generate a score matrix for potential loop closure pairs. Based on this score matrix, Gaussian smoothing and non-maximum suppression are applied to expand and refine the list of potential loop closure pairs. Finally, for each pair of frames, the registration algorithm for adjacent frames, as previously described, is applied. During registration, if more than 25 SIFT key points are matched, the feature tracks are merged. This conservative threshold ensures higher accuracy and reduces the likelihood of incorrect loop closures.

By aligning the frame pairs, the correspondence of the inliers can be obtained, which will be used in the bundle adjustment algorithm mentioned in the content of multi-camera angle calibration and fusion below. The estimated camera pose relative transformation can be used for the subsequent coordinate transformation, thereby generating a 3D point cloud model of the scene through data fusion.

2. Calculation of 3D Points: The ultimate goal of reconstruction is to generate a collection of 3D points, also known as a collection of voxels, or a point cloud file. A point cloud is a set of numerous data points that represent the spatial

distribution and surface characteristics of an object within the same spatial coordinate system. There are primarily two principles for obtaining point cloud data: laser measurement and laser photography. The laser measurement principle provides the 3D coordinate information (XYZ) of the object and the intensity of the laser reflection. In contrast, the laser photography principle provides both the 3D coordinate information (XYZ) and color information (RGB). Combining these two methods allows for the simultaneous acquisition of complete 3D coordinate data (XYZ), laser intensity information, and color information (RGB). The data from the point cloud includes spatial resolution, point accuracy, surface normal vectors, and more. After collecting and computing the 3D coordinates of each sampling point on the object's surface, a vast collection of 3D points is obtained, referred to as a "point cloud."

The camera collects information in the world coordinate system and converts it into its own stored image pixel coordinate system. Since the resolution of the two-dimensional image coordinate system is 640×480 , the corresponding X -axis coordinate range is $[0,640]$, and the Y -axis coordinate range is $[0,480]$. Through mathematical calculations, the X -axis coordinates and Y -axis coordinates in the image pixel coordinate system are converted into the X -axis coordinates and Y -axis coordinates in the world coordinate system with the Xtion sensor as the reference system (because the initial sensor coordinates coincide with the origin of the world coordinate system, so the camera coordinate system coincides with the world coordinate system). The schematic diagram of the image pixel coordinate system and the camera coordinate system with the Xtion sensor as the origin, which is also the world coordinate system, are shown in Figs. 6.14 and 6.15.

To achieve 3D reconstruction, it is necessary to complete the transformation of each point in the image in the two coordinate systems, and this transformation can be achieved through the following two formulas:

$$3D.X = (2D.X - cx_z) \times 3D.Z \times \frac{1}{f_{x_z}} \quad (6.3)$$

$$3D.Y = (2D.Y - cy_z) \times 3D.Z \times \frac{1}{f_{y_z}} \quad (6.4)$$

Fig. 6.14 Image pixel coordinate system

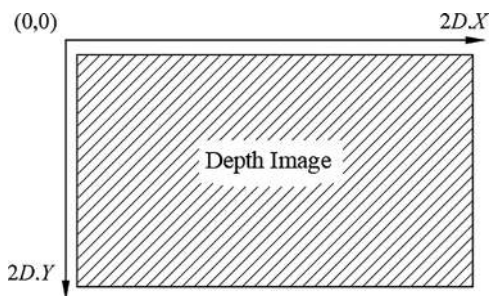
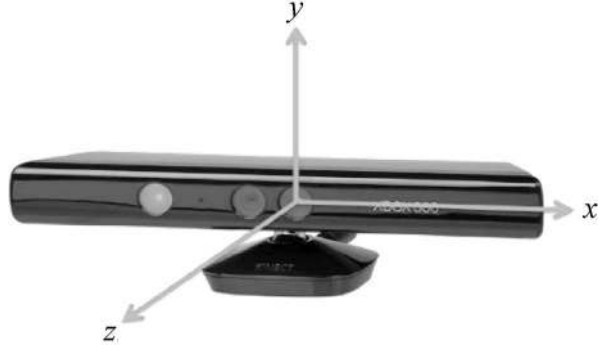


Fig. 6.15 World coordinate system with Xtion as the origin



In the formula, $3D.X$, $3D.Y$, and $3D.Z$ represent the X -axes, Y -axes, and Z -axes in the world coordinate system with the Xtion sensor as the origin, and $2D.X$ and $2D.Y$ represent the X -axes and Y -axes in the image pixel coordinate system. f_{x_z} and f_{y_z} represent the focal length of the depth sensor. cx_z and cy_z correspond to the X and Y coordinates of the camera center. In this project, because it corresponds to the center of the 2D image, its coordinates are $X = 320$, $Y = 240$. After the formula transformation, it can be converted into a 3D point in the world coordinates.

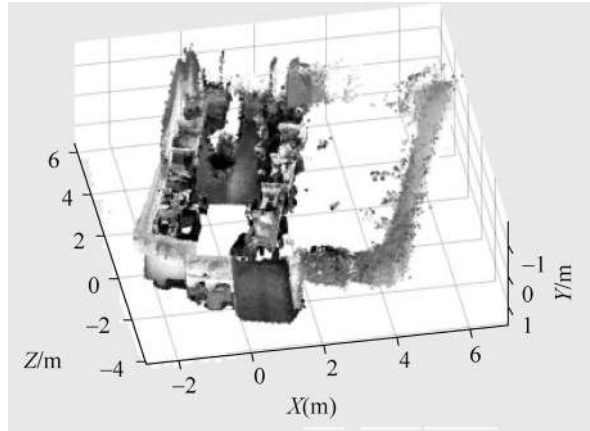
3. 3D Scene Modeling: With the camera pose, the world coordinates of the pixel points can be calculated through relative transformation, and the 3D scene model can be constructed through the point coordinates of the world coordinate system.
 - (a) 3D scene point cloud construction: Through the above-mentioned alignment of neighboring frames, all continuous frames can be aligned to the initial frame, thereby calculating the initial pose of the camera. For each frame, record the position of the robot, the rotation angle, and the transformation matrix of the camera. For any RGB-D image, first, convert it into a 3D point cloud in the camera coordinate system through the depth value of the pixel point and its (x, y) coordinates in the picture, according to the imaging parameters of the camera itself, and then calculate the world coordinate system according to the recorded parameters. Under, the 3D coordinates of this pixel point. The calculation formula is as follows:

$$X' = R_y(\theta_n)(R_c X + t_c) + [x_n, 0, z_n]^T \quad (6.5)$$

Here, (x_n, z_n) represents the distance of the robot on the ground plane. Since the height of the robot in the y -direction is constant, the y -coordinate of the horizontal plane of the robot's height is set to 0. $R_y(\theta_n)$ is the angle of the robot's rotation relative to the y -axis, and (R_c, t_c) is the rotation and translation matrix of the RGB-D camera relative to the robot base.

- (b) Scene division: Collecting RGB-D data for the entire indoor scene results in an exceptionally large amount of data. Additionally, due to the complexity of indoor environments, the robot cannot complete a generalized scene scan

Fig. 6.16 Results of running on a single camera scanning sequence



by simply moving in a straight line. Therefore, in this project, the scene is not only manually divided into sub-scenes during the robot's scanning process but is also further decomposed within the scene construction algorithm. Specifically, every 1000 frames of RGB-D sequences are divided into a sub-sequence, which accelerates the reconstruction process of long sequences. For the 3D point clouds constructed from adjacent sub-sequences, the RGB-D images from the two adjacent frames in each sub-sequence are coherent and overlap. This overlap allows for alignment using the Iterative Closest Point (ICP) algorithm, facilitating the merging of point clouds, as illustrated in Fig. 6.16.

6.4.2.3 Multi-Camera Perspective Calibration and Fusion

The calibration and fusion of multi-camera perspectives involve addressing two main issues. First, after obtaining the sub-scenes from each camera's field of view, it is necessary to align the RGB-D sequences collected by cameras with different orientations and elevation angles through camera calibration. This ensures that different RGB-D sequences can construct a unified point cloud model of the scene within the same world coordinate system. Second, in a complex indoor environment, it is impractical for the robot to scan the entire space along a straight-line path. Therefore, as previously mentioned, the scene is manually divided into several sub-scenes before scanning. For each sub-scene scanned by the robot, it is essential to stitch the point clouds together to expand the constructed scene range, ultimately obtaining a 3D model representation of the entire indoor environment.

For the first problem, this project employs the bundle adjustment method to globally optimize the camera poses initialized in the previous stage. The algorithm corrects the camera's rigid transformation by minimizing the objective function of bundle adjustment. For stitching the point clouds of divided scenes, a classic ICP-based point cloud stitching approach is used. This algorithm achieves registration

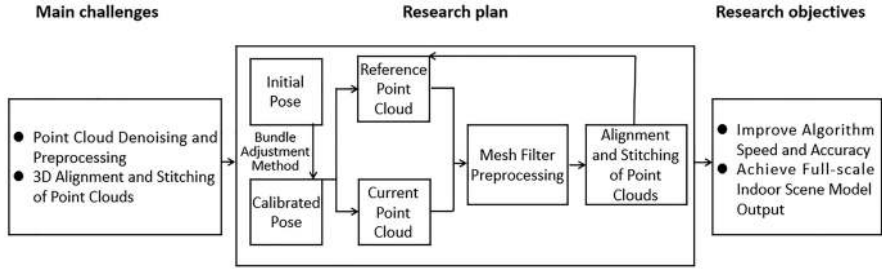


Fig. 6.17 Research route of multi-camera perspective calibration and fusion

by minimizing the differences between point clouds. However, since the point clouds generated during reconstruction can be coarse, it reduces calibration accuracy. To address this, the project first applies a grid filter for point cloud preprocessing. The grid filter divides both the reference point cloud and the current point cloud into numerous small cubes, and each cube is merged into a single point by averaging the spatial coordinates of the points within it. Next, the two-point clouds, after grid filtering, estimate the rigid transformation using ICP. Finally, the current point cloud is aligned to the overlapping area of the reference point cloud through rigid transformation. The point cloud is then merged onto the reference point cloud’s coordinate system, serving as the new reference point cloud for stitching the next input point cloud. The research route and goals for multi-camera perspective calibration and fusion are shown in Fig. 6.17.

Camera Pose Calibration

1. Initial calibration of multiple cameras. In the data-driven SFM algorithm, the camera pose corresponding to each frame in the RGB-D sequences collected by four cameras can be obtained, where the initial camera pose is $[E \ 0]$, indicating that the rotation matrix is a unit matrix and the translation parameters are all 0. However, due to the different orientations and angles of different cameras, their camera coordinate systems are also different. Therefore, it is necessary to calibrate the initial pose of the camera to ensure that each frame of each camera’s RGB-D can be calibrated to the same coordinate system through relative transformation.

In the system practiced in this project, four RGB-D cameras are placed on the flat top of the TurtleBot2 robot, and the height in the direction of the center of gravity is the same. Therefore, in the camera coordinates of the four cameras, the y-direction coordinates do not need to be calibrated. For the camera array composed of four RGB-D cameras, the four cameras face different angles, the diagonals of the camera center are perpendicular to each other, and the tilt angles of the four cameras are, respectively, 5° upward tilt, 10° downward tilt, 25° downward tilt, and 40° downward tilt. In this way, the rotation matrix of adjacent

cameras can be obtained by rotating 90° around the y -axis and 15° around the x -axis, and the translation matrix can be achieved by translating half of the diagonal in the x and z directions to achieve the rigid transformation of the two adjacent cameras.

This project takes one of the RGB-D cameras as the reference camera, sets the camera pose of its first frame as $[E \ O]$ (the rotation matrix is a unit matrix, and the translation matrix is 0), and the other cameras calculate the spatial transformation relative to the reference camera through the spatial relationship in the camera matrix, and update the camera pose of the remaining frames. Through the above initial calibration of the camera RGB-D sequence, each frame can be transformed to the coordinate system of the reference camera's initial frame through the corresponding camera pose transformation matrix. The accuracy of these relative pose transformations is not high and needs to be further globally optimized by the bundle adjustment algorithm.

2. Bundle adjustment method. The bundle adjustment method is a method of analytical photogrammetry. In the analysis, the collinearity equation is used as a mathematical model to calculate and provide an approximate solution by linearizing the observed values of the image plane coordinates of the image points, and finally, iteratively approaching the best value to achieve the purpose of global optimization.

Although the camera pose for each RGB-D frame can be obtained in the previous stage, and data fusion through these poses can generate a 3D point cloud, the results are often very inaccurate. This is because, after reconstruction, the 3D point cloud needs to be projected back into an image and must remain as consistent as possible with the original RGB-D image. Therefore, bundle adjustment is required for global optimization to ensure that the projected image points are globally as close as possible to the observed values.

Since the RGB-D sequence used in this project practice is continuous, the correspondence of SIFT features exists in multiple frames. By linking the key points in the same position in each frame, longer feature trajectories can be obtained. The objective function for iteratively obtaining global minimization by reprojecting the 3D point cloud is as follows:

$$\min \sum_c \sum_{p \in P(c)} \left(\|\tilde{x}_p^c - K[R_c | t_c] X_p\|^2 + \lambda \|\tilde{X}_p^c - [R_c | t_c] X_p\| \right)^2 \quad (6.6)$$

where K is the intrinsic matrix of the RGB-D camera, the intrinsic matrix parameters of the same camera are the same, R_c and t_c , respectively, represent the rotation transformation matrix and translation transformation vector of the camera pose corresponding to the frame c . X_p is the coordinate of the 3D point visible under this camera pose, and $\tilde{x}_p^c$ and $\tilde{X}_p^c$ are the 2D pixels observed in the camera coordinate system and the coordinates of the corresponding 3D points.

By globally optimizing each camera pose for every frame using the bundle adjustment algorithm, the generated point cloud can be made more closely resemble the point cloud observed after projection. This enhances the ability of

the calibrated multi-camera poses to reflect the transformation between the camera coordinate system and the world coordinate system on a global scale. The calibrated and optimized camera poses can then be used for data fusion to generate scene point clouds, as described in the previous section. In this project, the poses of the four cameras are iteratively optimized together using bundle adjustment after calibration. Consequently, a globally optimized scene point cloud model that fuses the perspectives of the four cameras can be generated.

Point Cloud Alignment Method Based on ICP

1. Noise reduction and preprocessing. Due to the rough nature of the point cloud generated during the reconstruction process, the quality of point cloud registration depends on the noise in the data and the initialization settings of the ICP algorithm. Therefore, some preprocessing is necessary to filter out noisy data. This project uses a grid filter for noise reduction and preprocessing. The grid filter sets a bounding box of a certain size and divides the entire point cloud space into countless small squares of the same size, using a grid side length of 10 cm in this project. Each square contains a varying number of 3D points. The average coordinates (x , y , z) of all 3D points in the square are taken as a new 3D point, while the other points in the square are discarded. By using the grid filter, points that are close in local space can be merged into one, reducing the number of points in the point cloud and also lowering the proportion of noise points. Compared with the unprocessed point cloud, this not only speeds up the point cloud registration process but also improves the accuracy of the ICP algorithm.
2. ICP algorithm and point cloud alignment. The ICP algorithm is an algorithm used to minimize the difference between two-point clouds, usually used in 2D and 3D surface reconstruction under different viewing angle scans. In the ICP algorithm of this project practice, the point cloud processed by the grid filter is used as input, with the first point cloud as the reference point cloud, and the next point cloud as the current point cloud. First, for each point in the current point cloud, find the nearest point in the reference point cloud; second, use the mean square difference value function to estimate the combination of rotation and translation, to optimally align the current point to the nearest point found in the reference point cloud in the previous step; Finally, use the estimated rotation and translation to transform the points in the current point cloud, and continue to iterate.

Scene-Based Point Cloud Stitching

Through such an ICP algorithm, the rigid transformation from the current point cloud to the reference point cloud can be finally obtained. Through this rigid transformation, the current point cloud is transformed into the reference point cloud coordinates, which can complete the alignment of the point cloud.

After aligning the two-point clouds, it is necessary to fuse the overlapping areas of the two-point clouds, and then complete the stitching of the two-point clouds in the reference point cloud coordinate system. The same grid filter is used to process the overlapping area. The grid bounding box side length set in this project practice is 2 cm, dividing the overlapping area into countless small squares with a side length of 2 cm, and all points in the square are merged into one point (the spatial coordinates are the average of all point coordinates). Through the processing of the grid filter, the overlapping area can be fused to form a stitched point cloud. This stitched point cloud becomes the new reference point cloud, waiting for the alignment and stitching of the next current point cloud. Through the stitching of multiple point clouds, a small range of scene point clouds can be fused into a larger range of 3D models, thereby realizing the construction of a large range of indoor scenes.

6.4.2.4 Experimental Results

Using the camera pose after global optimization for data fusion, a more accurate scene model can be obtained. The comparison before and after optimization is shown in Fig. 6.18.

As can be seen from the figure, the bundle adjustment method can significantly optimize the global camera pose, thereby obtaining a more accurate 3D model.

After obtaining the models for each sub-scene, the system needs to run point cloud alignment and stitching algorithms offline to extend the model's range.

The algorithm starts with preprocessing the point cloud data with a mesh filter to improve the speed and accuracy of registration. For display convenience, this project practice only shows the stitching process of two-point clouds. The

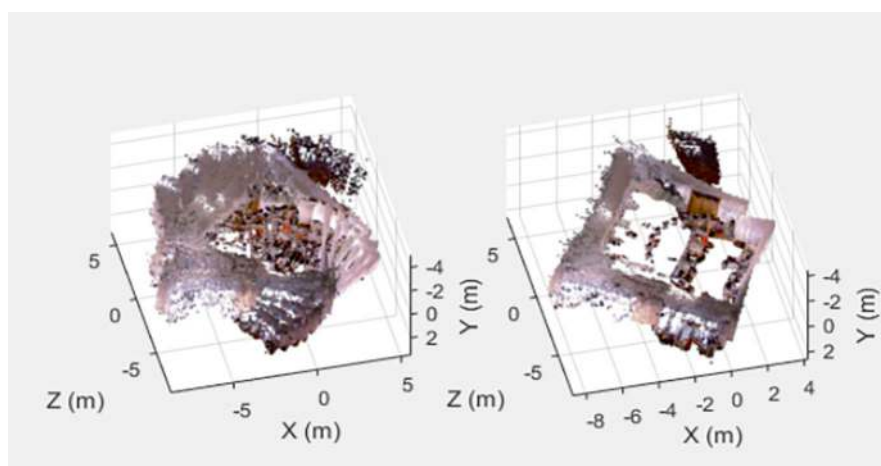


Fig. 6.18 Comparison of effects before and after bundle adjustment optimization. (a) Before calibration; (b) After using bundle adjustment

comparison before and after point cloud preprocessing is shown in Fig. 6.19. Then, the processed point clouds are aligned using ICP. In this project practice, the alignment results of two-point clouds taken as reference coordinate systems are shown in Fig. 6.20. The final stitching result of the two-point clouds can be seen in Fig. 6.21.

As can be seen from Fig. 6.18, the system can perform noise reduction on the initial point cloud with a lot of noise and reduce the number of sample points. The processed point cloud can complete the alignment algorithm well. After the point cloud alignment, in order to ensure consistency with the original scene, the initial input two-point clouds (i.e., unprocessed point clouds) are fused to finally obtain the stitching result of the two-point clouds (Fig. 6.21).

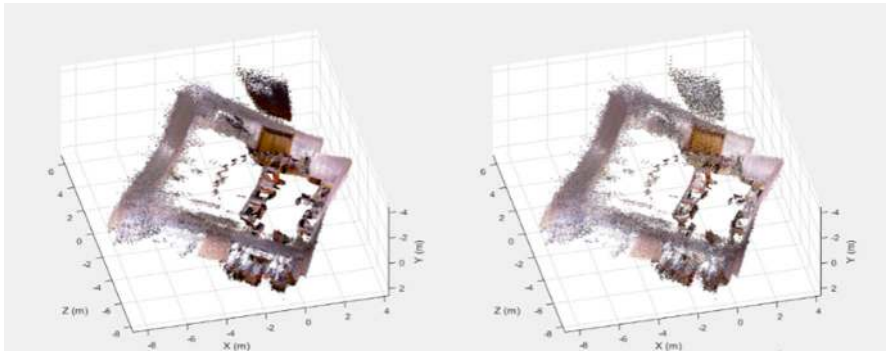


Fig. 6.19 Comparison before and after preprocessing with a mesh filter. (a) Before preprocessing; (b) After preprocessing

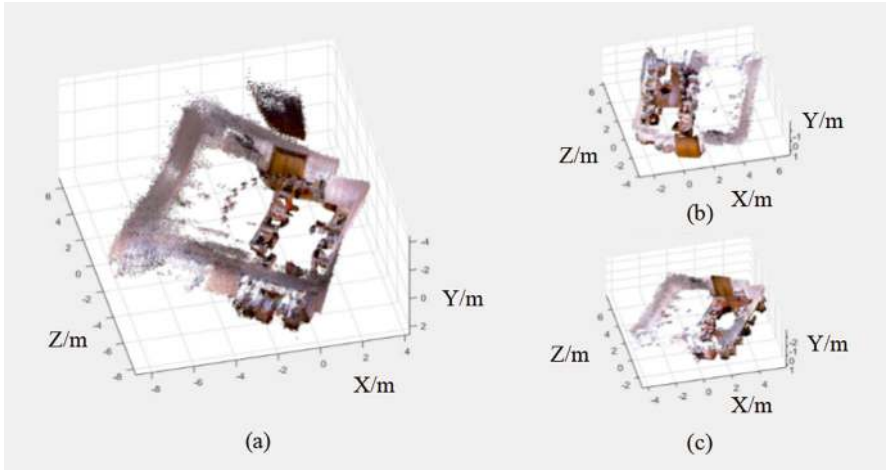
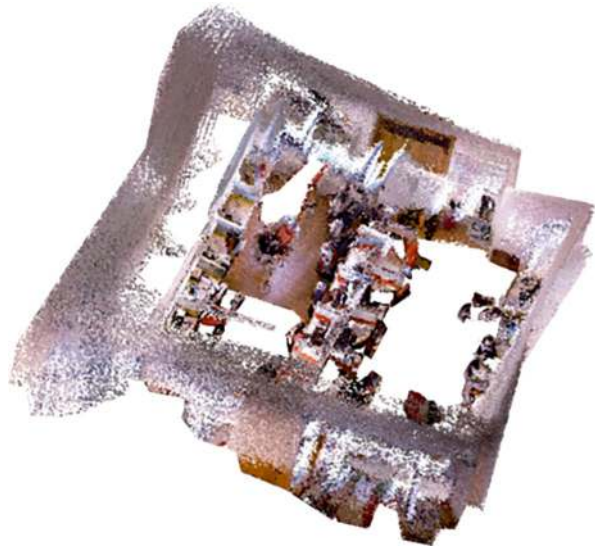


Fig. 6.20 Point cloud alignment effect. (a) Point cloud 1 (reference point cloud); (b) Point cloud 2 (current point cloud) before alignment; (c) Point cloud 2 (current point cloud) after alignment

Fig. 6.21 Stitching result of two-point clouds



6.4.3 Scene Labeling Technique Based on RGB-D Sequence

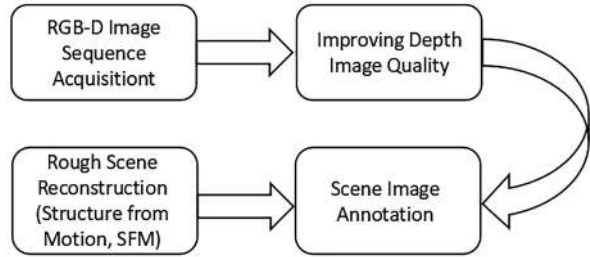
Traditional scene labeling techniques are divided into manual labeling and automatic labeling. Manual labeling is highly reliable, but its efficiency is often low for large-scale scene labeling; Although automatic labeling has improved efficiency, it lacks accuracy and reliability, which adds many difficulties to the utilization of scene information; Moreover, traditional scene labeling techniques often require collection and reconstruction processes to be performed on a single computer, which places high demands on the computer's storage and computing speed.

Therefore, a scene cloud labeling technique based on RGB-D sequence was proposed, and a web-based cloud labeling tool was developed. First, the labeling tool is used to manually label certain scenes. Then, the frame transfer algorithm is used and some reconstruction information is used to transfer the completed labels to other frames to reduce repeated labeling and improve labeling efficiency; Finally, the labeling information is used to improve the quality of scene reconstruction and the semantic mapping from 2D to 3D.

6.4.3.1 Research Plan

As shown in Fig. 6.22, first, to collect depth video frames, an Asus Xtion Pro Live depth sensor is connected to a lower-level machine. Video is captured using OpenNI at a resolution of 640×480 with a sampling rate of 30 fps. The default factory calibration of the depth camera is used for the registration of the depth map and color map. Only indoor scenes need to be scanned since the Asus Xtion Pro Live depth camera cannot operate under direct sunlight. Each operator is required to thoroughly

Fig. 6.22 Scene labeling research plan



walk through every part of the entire space, carefully scanning every room, including the floor, walls, and all objects.

The original depth maps collected usually contain a lot of noise and holes, which can affect the formation of polygonal bounding boundaries during object labeling, thereby reducing the accuracy of the frame transfer algorithm. To fill these holes and improve the quality of the depth map, the current mainstream approach is self-cross bilateral filtering, which produces visually pleasing depth maps but introduces many artificial artifacts. In contrast, by using the truncated signed distance function (TSDF) to voxelize space, and leveraging the camera position information obtained earlier, depth maps can be accumulated from nearby frames (up to 40 of the closest frames). By using only temporally adjacent frames, reliable information about the local camera position can usually be easily obtained. Finally, ray casting is used to generate reliable depth maps for each frame.

Then, the SFM algorithm and the collected RGB-D depth sequence are used for preliminary scene reconstruction. The reconstruction process will generate many errors. However, many of these errors are essentially long-interval, and small errors will accumulate over time, often resulting in more accurate reconstruction of some sub-sequences. A web-based marking tool has been developed to take advantage of this, which can manually annotate the 3D model using this model and the RGB-D depth sequence. These marks will be used to improve the quality of reconstruction in subsequent processes.

Depth Map Quality Improvement Algorithm

The original depth maps obtained after collection often contain noise and holes, which can cause significant interference when manually marking objects. To eliminate this interference, it is necessary to enhance the quality of the depth map to reduce the occurrence of noise and holes. Based on a review of relevant literature, two main algorithms are currently recognized as effective for improving the quality of depth maps: one is the cross bilateral filtering method, which has the advantage of producing smoother depth maps but the drawback of introducing noticeable errors; the other is the TSDF method used by the Princeton Computer Vision Group, which avoids introducing artificial errors and also yields good results in processed depth maps.

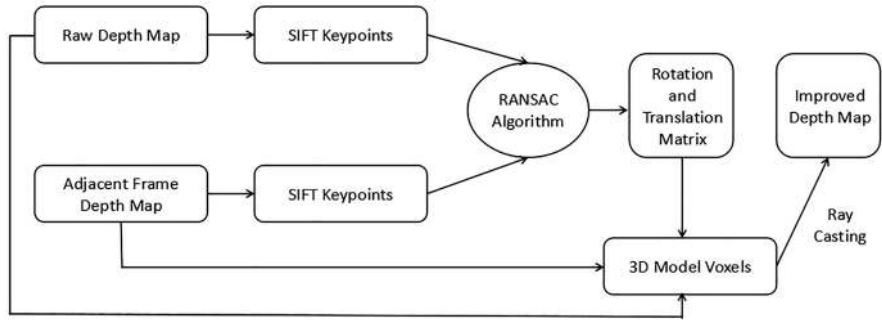


Fig. 6.23 Process of improving depth map quality

Basic idea of the TSDF method: The generation of noise and holes is random. An object in the depth map of a certain frame may have some noise and holes, but the same object in other nearby frames may not contain these imperfections. By leveraging information from nearby frames, it is possible to eliminate the noise and holes in the current frame, thereby improving the quality of the depth map. The general process for improving the quality of the depth map is shown in Fig. 6.23.

The algorithm for improving the quality of any one frame depth map A is as follows:

- Step 1: Obtain the timestamp information of depth map A.
- Step 2: Use the timestamp information of depth map A to obtain the depth maps of N (take 20) temporally adjacent frames.
- Step 3: Use the SIFT algorithm to process depth map A and the depth maps of adjacent frames to obtain their respective SIFT key points.
- Step 4: Use the rotation and translation matrices obtained after reconstruction to obtain the 3D coordinates of each SIFT key point in the reconstructed model.
- Step 5: Divide the 3D model into small cubes (with a minimum length of no <0.4 mm).
- Step 6: Accumulate the depth information of each depth map onto the small cubes.
- Step 7: Using the ray projection algorithm, the improved depth map is obtained using the camera position information from depth map A.

Research on Scene Marking and Frame Transfer Algorithm

After improving the quality, high-quality depth maps can be obtained, which can be used with 3D reconstruction to obtain a rough 3D model for manual marking.

If none of the depth maps have been manually annotated, then manual annotation is required for any frame. First, for an object B in the selected frame, you need to manually use the mouse to draw its approximate closed contour line. The process of drawing the contour line is essentially the process of selecting key points along the object’s boundary. The coordinates of these selected key points are two-dimensional

and require further processing before they can be utilized. Each two-dimensional key point is mapped to the 3D scene data through calculations involving the camera position and the transformation matrix.

If there are already annotated depth maps, then when annotating a new frame of the depth map, you should first obtain the two keyframes that are closest in time to it. Each of these keyframes has been annotated and has a corresponding 3D polygonal region in the 3D model. The 3D polygonal region of each object annotated on each key frame is then projected onto the unmarked frame of the depth map according to the camera position information calculated during the reconstruction process, forming a two-dimensional polygonal boundary. The same strategy is applied to the other keyframe. In this way, the annotations on the depth map of the manually annotated keyframes can be automatically transferred to the unmarked depth map, significantly improving the efficiency of manual annotation.

However, due to numerous errors or inaccuracies in the camera position information generated during the 3D model reconstruction process, the transfer algorithm may sometimes deviate. For instance, the resulting 2D polygon boundary might significantly misalign with the object boundary on the depth map that needs to be annotated. To address this issue, a conflict list should be established, with its initial state being empty. When you are unsatisfied with the result produced by the frame transfer algorithm and make adjustments, you should first identify the original frame from which the annotation result was transferred and then add that frame to the conflict list. Consequently, when the frame transfer algorithm is executed, the frames in the conflict list will no longer be considered as keyframes.

A typical result of the frame transfer algorithm is shown in Fig. 6.24. In Fig. 6.24a, a chair is marked as a keyframe, and in Fig. 6.24b, a trash can is also marked as a keyframe. The frame in Fig. 6.24c is between these two frames, and no annotation was made for Fig. 6.24c. The annotations for the chair and trash can in Fig. 6.24c were transferred from the two keyframes before and after. It can be seen that although the frame transfer algorithm has some deviations, its annotation effect meets the requirements.

6.4.3.2 Experimental Results

Figure 6.25 demonstrates the correction of the frame transfer algorithm. As shown in Fig. 6.25a, it is a manually annotated image of a pillow, and Fig. 6.25b shows the result of marking using the frame transfer algorithm after several frames. It can be clearly seen that there is a large deviation. Figure 6.25c is the result after manual correction. Figure 6.25d is the result of marking a new frame after correction by the frame transfer algorithm. It can be seen that because the marking in Fig. 6.25a was added to the conflict list, Fig. 6.25c no longer uses it as the source of frame transfer, but uses the marking in Fig. 6.25d, and the effect is also enhanced. In summary, Fig. 6.26 is the basic marking process.

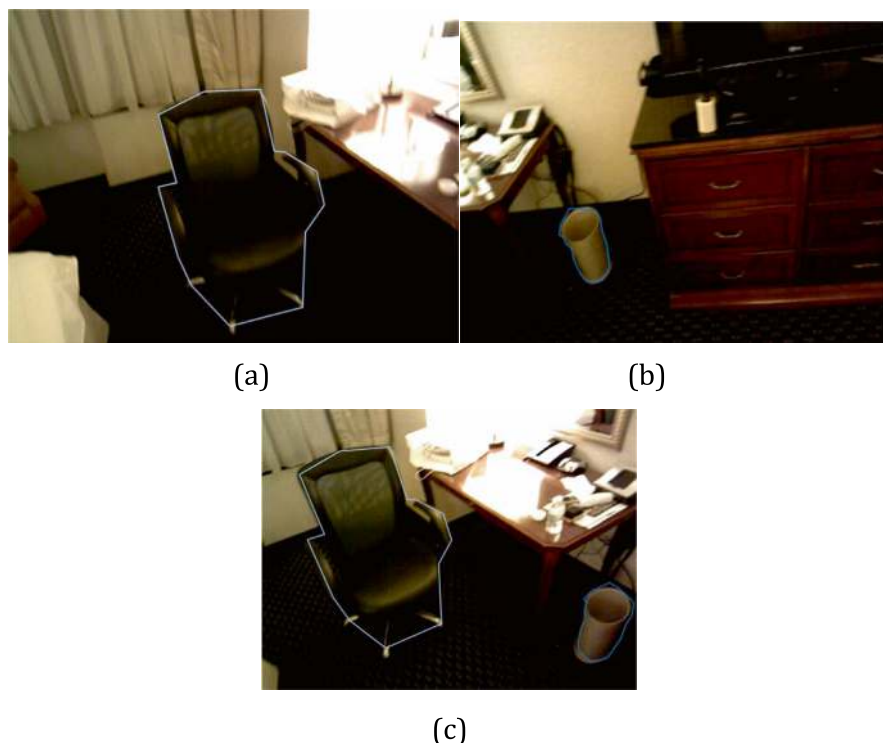


Fig. 6.24 Typical experimental results of the frame transfer algorithm. (a) Marking of the chair; (b) Marking of the trash can; (c) Effect of frame transfer algorithm

For each 2D key point, it is mapped to the 3D scene data through the calculation of the camera position and the transformation matrix. A 3D semantic annotation scene is generated, as shown in Fig. 6.27.

6.4.4 Platform Design

6.4.4.1 Overall Architecture of the Simulation Platform

In order to obtain the RGB-D sequence, it is necessary to use the depth camera Xtion Pro Live for information collection. The collected data needs to be sent to a remote machine via the network. During the collection process, it is necessary to control the robot carrying the depth camera to walk along the predetermined route, which requires a module for controlling the robot. After data transmission is complete, a 3D reconstruction program must be run to use the collected RGB-D sequence to reconstruct the 3D model and generate camera position information. Then, the rough 3D model, position information, and RGB-D sequence need to be used to

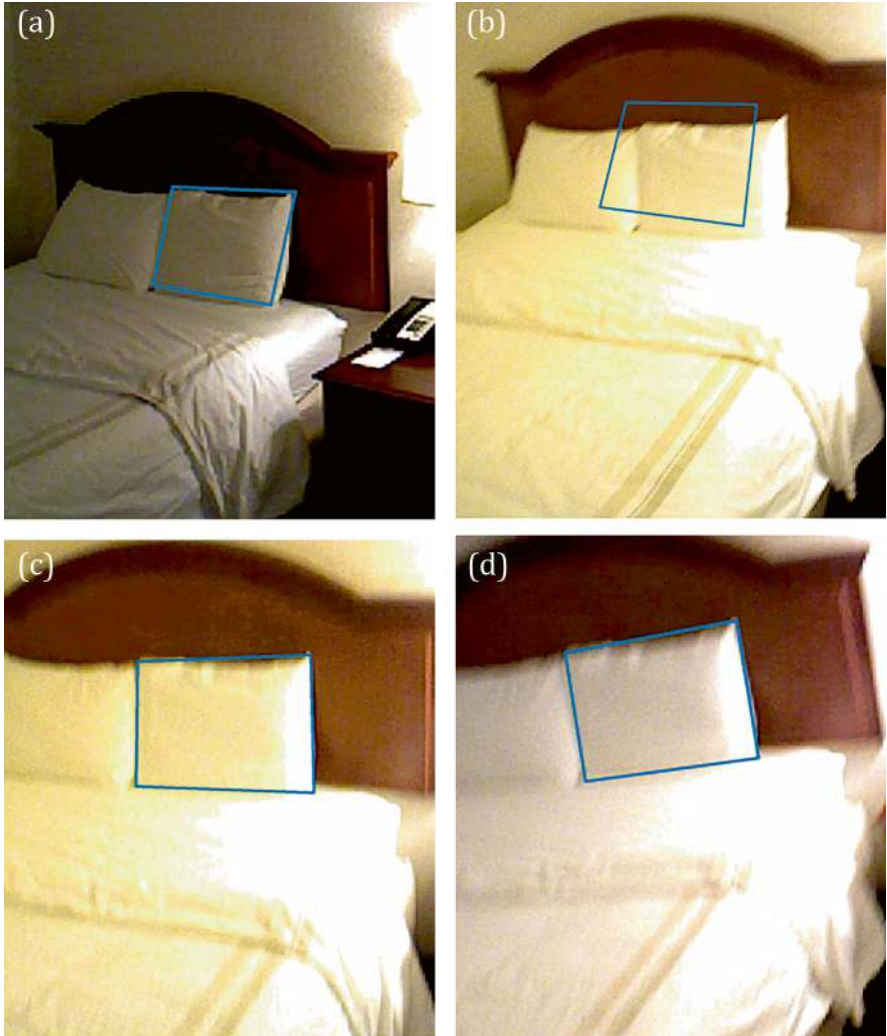


Fig. 6.25 Correction of the frame transfer algorithm. (a) First marking of the target object; (b) Result after several frame transfers; (c) Result after correction; (d) Frame transfer result after correction

manually annotate the scanned scene. Since a web-based cloud annotation tool has already been developed, it is only necessary to set up a web server to return the corresponding data when requested by the browser. Each simulation platform requires a settings module to configure various parameters related to the function.

Based on the above ideas, the simulation platform includes information collection, 3D scene reconstruction, scene cloud annotation, and setting modules. The information collection module includes real-time display saving and real-time path display sub-modules. The overall architecture is shown in Fig. 6.28.

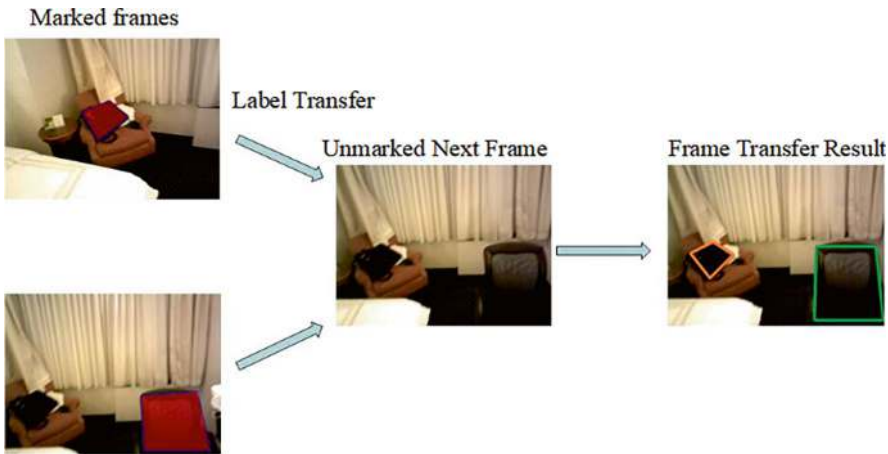


Fig. 6.26 Marking process



Fig. 6.27 Semantic annotation results

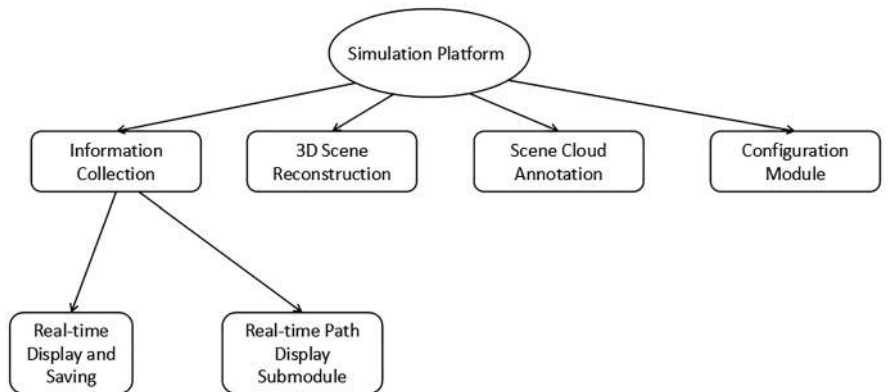


Fig. 6.28 Overall architecture of the simulation platform

6.4.4.2 Division of Simulation Platform Modules

The simulation platform mainly consists of information collection, 3D scene reconstruction, scene cloud annotation, and setting modules.

3D Collection

The 3D collection part needs to accomplish two main functions: one is to receive the information collected by the four depth cameras, display, and store it; the other is to implement the robot control functionality. To achieve these functions, a socket is used to establish a connection between the simulation platform and the collection program, with the socket port obtained from the settings. Since using a socket to receive images causes blocking, if all image displays are handled in the main thread (which displays the graphical user interface, GUI), the GUI may become unresponsive due to this blocking. Therefore, multithreading technology is employed. A total of seven threads are created: one for handling user operations (the GUI thread), and the remaining six threads each establish a socket connection to receive information.

In order to control the robot, a socket is used to directly send control command strings. The strings are “start,” “forward,” and “turn,” which, respectively, represent start collection, move forward once, and rotate clockwise once. The distance for moving forward once is 0.3 m, and the angle for rotating clockwise once is 5° . This has been preset in the collection program. By combining these two types of movements, the robot can be made to walk along any path. In this way, after the collection program receives the command, it will execute the corresponding command, allowing the robot to walk along the predetermined route and collect.

Implementing the depth camera to receive, display, and store images is much more complex. First, the received information must include both the name and the image itself for each picture. Additionally, since the size of the picture is not fixed, it cannot be read using a fixed length of bytes. To address this, a sending structure is established with three components: the picture name, picture size, and the picture itself. The picture name is composed of a 7-digit code, a hyphen (“-”), and a 12-digit timestamp, resulting in a fixed-length 21-byte string. Thus, the receiving end only needs to receive a 21-byte string. The picture size is represented by a 4-byte integer. After determining the picture size, the sending end transmits this size to the simulation platform. The platform then reads the 4-byte integer to obtain the picture size. With this information, the simulation platform reads the picture from the receiving buffer. Consequently, the process of receiving a complete picture on the simulation platform involves: reading a 21-byte string as the picture name, reading a 4-byte integer for the picture size, and then using the picture size to read the entire image data.

The received picture is just a string of bytes without any meaning. The conversion parameters for the color image are $640 \times 480 \times 3$, and the conversion parameters for the depth image are $640 \times 480 \times 1$. Then, the picture needs to be displayed at a specific location in the window. The picture is received by a sub-thread, and the

sub-thread is also responsible for refreshing the display of the picture. Due to security reasons, sub-threads are not allowed to perform related display operations. To solve this problem, the “signal-slot” mechanism between threads is used. Firstly, a slot function is defined in the main thread to receive signals from the sub-thread and perform corresponding operations. Second, after the sub-thread receives a complete picture, it stores it in a fixed path folder and sends the string form of the picture path as a parameter to the main thread. The main thread extracts the picture path parameter after receiving the signal. Finally, it loads the picture from the path and displays it at a fixed location in the window.

Through the above methods, the storage and display of pictures can be solved. Since the real-time path information of the robot is also in the form of pictures, the implementation method is the same, so it will not be repeated here.

Scene Reconstruction

The core code for scene reconstruction is implemented using MATLAB software. Therefore, it only needs to call the MATLAB program in the simulation platform and display the processing results. To solve this problem, a new sub-thread is started. This thread will call the MATLAB engine to execute the corresponding program. The MATLAB script path and the folder path where the color and depth images used in the reconstruction process are located are passed to the sub-thread as parameters.

During the process of calling the MATLAB engine to execute the program, if a reconstruction error occurs, this error will be captured and displayed. Otherwise, after the program is executed, the reconstructed 3D point cloud model will be displayed at a specified location.

Scene Marking

The scene marking part requires running a web server. The current web server options mainly include Apache and nginx. The Apache server uses a synchronous multi-process model, where each connection corresponds to a separate process. In contrast, the nginx server employs an asynchronous model, allowing multiple connections to be handled by a single process. Compared to Apache, nginx is lightweight, straightforward, easy to configure, and stable. It is also particularly effective at handling static pages. Therefore, nginx is chosen as the web server. All resources generated earlier (including the RGB-D sequences and the 3D models created during the reconstruction process) are made available under nginx as uniform resource locators (URLs). When a URL request is sent from the browser, the nginx server responds to the request and returns the requested resources. After marking is completed on the browser side, the results are sent back to the server, thereby completing the marking task.

Setting Module

The setting module is an indispensable part of every simulation platform. The setting part of the simulation platform is mainly used to set various ports when establishing a socket connection. If the port format set is incorrect or the port is occupied, the simulation platform will give a prompt, and then reset the port.

6.4.4.3 Simulation Platform Operation Process

Figure 6.29 illustrates a more technically detailed architecture diagram, which explains the operational flow of the simulation platform. In the diagram, Rob represents the robot, which is equipped with four Asus Xtion Pro Live depth cameras and carries a lower-level machine. This lower-level machine runs ROS and is responsible for powering the four-depth cameras and the robot. The four depth cameras and the robot are connected to the lower-level machine via USB interfaces. The lower-level machine controls the robot's movement and collects and transmits depth maps using the cameras. The remote upper-level machine runs the pre-designed simulation platform, which handles receiving the transmitted depth maps, sending control

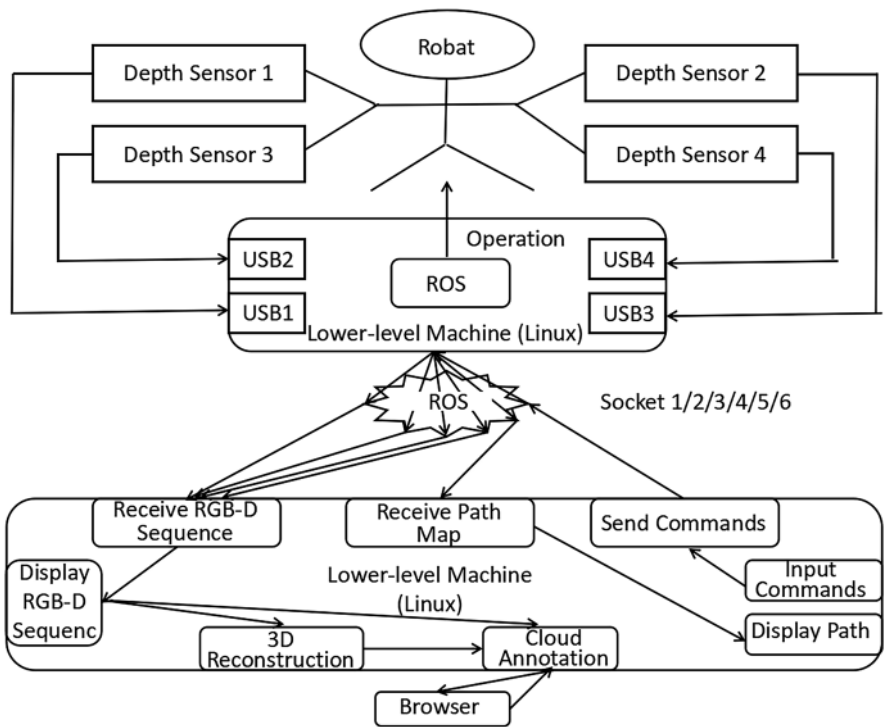


Fig. 6.29 Detailed architecture of the simulation platform

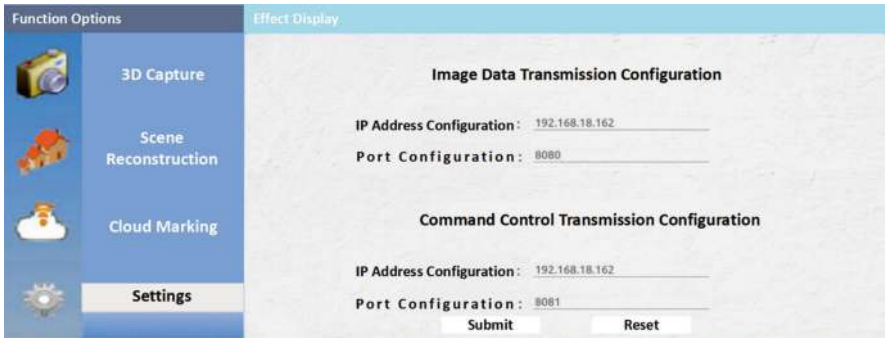


Fig. 6.30 Settings module

commands to the robot, running a web server, and responding to requests from remote browsers. Communication between the lower-level and upper-level machines is achieved via a wireless network, with a total of six socket connections established: four sockets for transmitting and receiving depth maps from the cameras, one socket for sending and receiving robot control commands, and one socket for transmitting and receiving images that show the robot’s real-time walking path.

The simulation platform runs on the upper-level machine. The interface is divided into two main parts: on the left side are the functional options, consisting of four modules: 3D Collection, Scene Reconstruction, Simulation Platform, and Settings. The right side displays the results window.

1. Step one: Set up the simulation. As shown in Fig. 6.30, the primary configuration involves setting the ports for the six sockets. First, click to run the 3D Collection module. This action will start six sub-threads, each of which will open one socket. The sockets then enter a listening state, with their ports obtained from the previous settings, defaulting to 0000. Finally, the upper-level machine will initiate connection requests to the six sockets. Once the connections are established, communication between the simulation platform and the collection program is successfully set up via the wireless network.
2. Step two: Control the robot using the sub-module that interacts with the robot. As shown in Fig. 6.31, click the “Start” command, and the simulation platform will send a command to the collection program to begin data collection. The collection program then activates the four depth cameras to gather data, which is sent to the simulation platform through four sockets. Upon receiving this data, the simulation platform displays it in eight windows, as illustrated in Figs. 6.32, 6.33, 6.34, and 6.35. Four of these windows within the total scene window show the RGB images (color images) captured by the four depth cameras, providing an overview of the scanned scene. The remaining four windows display the collected depth maps (depth images). Users can switch between displaying color images and depth images by clicking a button in the upper right corner. While displaying the data, the program differentiates it according to the depth cameras and stores it in separate folders. Additionally, the robot’s trajectory can be con-

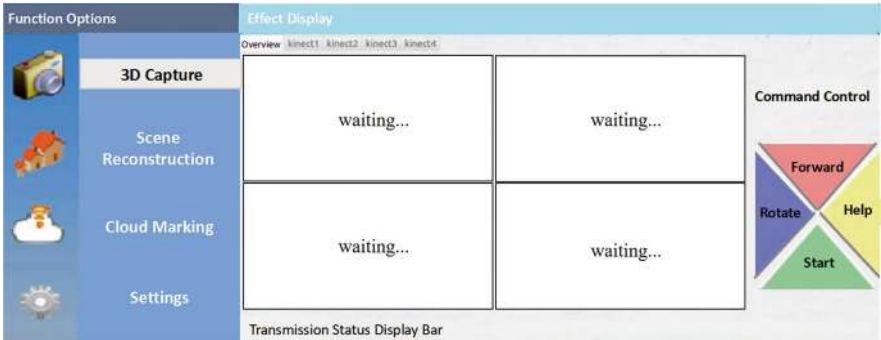


Fig. 6.31 Robot control module



Fig. 6.32 Display of collected information a (RGB image)



Fig. 6.33 Display of collected information a (depth image)

trolled through the interaction sub-module, allowing the robot to follow a pre-designed path for scanning, which enhances the quality of the collected data for subsequent processing.

As shown in Fig. 6.36, the robot’s movement is primarily controlled through two methods: clockwise rotation and straight-line walking. Each time the rota-

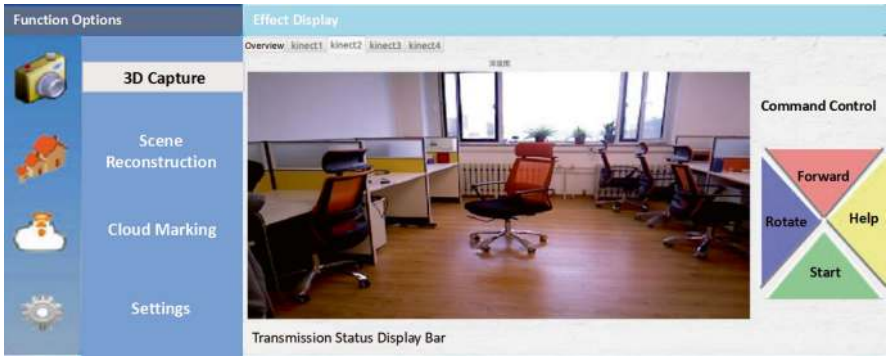


Fig. 6.34 Display of collected information b



Fig. 6.35 Display of collected information c (depth image)



Fig. 6.36 Display of 3D reconstruction results

tion button is clicked, the robot rotates 5° clockwise. Each time the straight-line walking button is clicked, the robot moves straight for 0.3 m. By combining these two control methods, the robot can be directed to follow any desired trajectory. Simultaneously, the collection program maps the robot's path based

- on its motion parameters and sends this data back to the system using a socket. The simulation platform then displays the path in the robot’s real-time path window. Observing this real-time path image helps in accurately tracking the robot’s movement, facilitating better control of the robot.
3. Step three: Click the “information collection” button again after the collection is completed. At this time, the simulation platform will send an end command to the collection program at the other end, causing the collection program to stop working; at the same time, the simulation platform will close all socket connections, thus completing the entire information collection task.
 4. Step four: Click the “Scene Reconstruction” button to run the 3D scene reconstruction module. This module uses the color and depth maps obtained from the information collection, along with the camera’s intrinsic matrix, to generate a rough 3D model and the camera position information for each image. The camera position information is stored in a JSON (JavaScript Object Notation) file. Due to the large computational workload, this process is handled in the cloud. As shown in Fig. 6.36, once completed, the simulation platform will display the generated 3D model in the window. The 3D model is a point cloud stored in PLY format.
 5. Step five: Run the scene marking module to manually annotate the 3D model reconstructed from the collected depth maps. Since the cloud-based annotation approach is employed, the marking task is distributed to other endpoints to reduce the computational load on the primary machine. Thus, it is sufficient to start a web server on the upper-level machine. As shown in Fig. 6.37, once the web server is running, it can handle browser requests from other endpoints and allocate the necessary resources for annotation, such as RGB-D sequences and camera position data. After receiving the resources and response, the browser on the other end performs manual annotation, as shown in Fig. 6.38, and then sends the annotated and processed information back to the web server.

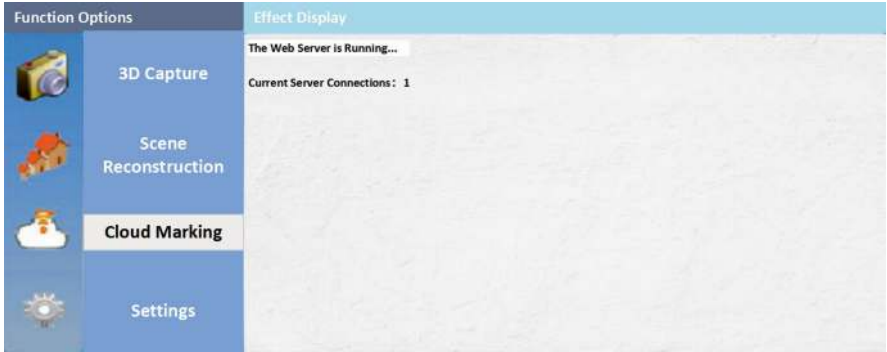


Fig. 6.37 Web server startup

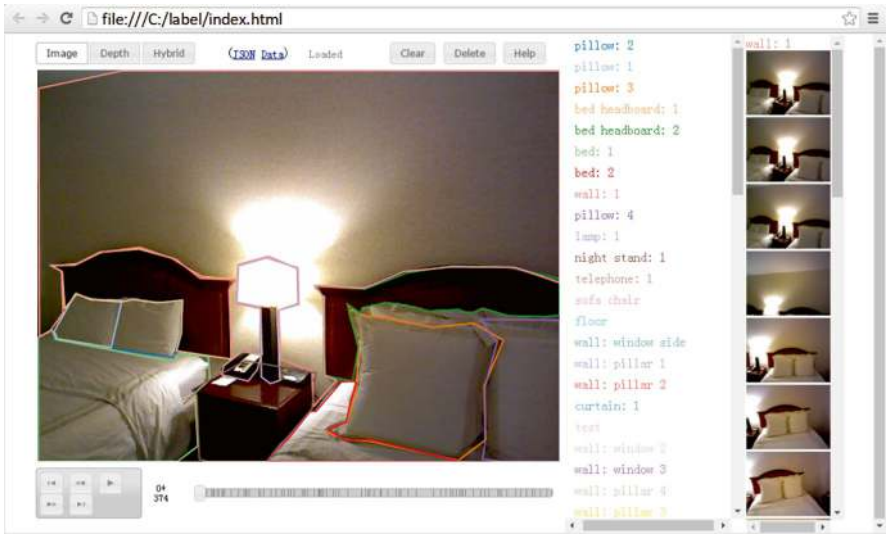


Fig. 6.38 Cloud annotation tool

6.5 Summary

The goal of this project is to extend the application of robot vision understanding methods to handle scenarios with potentially incomplete information. It aims to develop a framework for robotic scene understanding. By implementing this project, the robot will be able to achieve more precise autonomous scene comprehension and ultimately perform navigation and recognition tasks based on its visual understanding of the environment.

6.6 Discussion

In this project, 3D scene modeling and understanding for robots have been achieved. However, it is evident that in the initial data collection phase, manual control is required to move the robot or to use predefined paths for data collection. This approach results in insufficient system autonomy, and since data collection and scene modeling understanding are separate tasks (i.e., collect data first, then perform reconstruction), it leads to reduced flexibility and efficiency. Therefore, consider the following: How can we enable robots to perform real-time scene modeling and understanding while collecting data, and subsequently use this capability to complete the data collection process? The goal is to integrate data collection with scene modeling and understanding to achieve autonomous scene modeling and understanding.

References

1. NILSSON N J, AUTOMATON A M. An application of artificial intelligence techniques[C]// Proc of IJCAI, 1969.
2. TURK M A, MORGENTHALER D G, GREMBAN K D, et al. Video road-following for the autonomous land vehicle[C]. Proceedings of the 1987 IEEE international conference on robotics and automation, 1987, 4: 273–280.
3. TURK M A, MORGENTHALER D G, GREMBAN K D, et al. Vits-a vision system for autonomous land vehicle navigation[J]. IEEE transactions on pattern analysis and machine intelligence, 1988, 10(3): 342–361.
4. THORPE C, HEBERT M, KANADE T, et al. Toward autonomous driving-the cmu navlab.1. Perception[J]. IEEE Expert-Intelligent Systems & Their Applications, 1991, 6(4): 31–42.
5. THORPE C, HEBERT M, KANADE T, et al. Toward autonomous driving-the cmu navlab.2. Architecture and systems[J]. IEEE Expert-Intelligent Systems & Their Applications, 1991, 6(4): 44–52.
6. KALOGERAKIS E, HERTZMANN A, SINGH K. Learning 3D mesh segmentation and labeling[J]. Acm Transactions on Graphics, 2010, 29(4): 1–12.
7. SIDI O, Van KAICK O, KLEIMAN Y, et al. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering[J]. Acm Transactions on Graphics, 2011, 30(6): 1–9.
8. SUNKEL M, JANSEN S, WAND M, et al. Learning line features in 3D geometry[J]. Computer Graphics Forum, 2011, 30(2): 267–276.
9. XU K, HUANG H, SHI Y F, et al. Autoscaning for coupled scene reconstruction and proactive object analysis[J]. Acm Transactions on Graphics, 2015, 34(6): 1–14.
10. LAI K, BO L F, FOX D, et al. Unsupervised feature learning for 3D scene labeling[C]//IEEE international conference on robotics and automation. New York: IEEE, 2014: 3050–3057.
11. AYDEMIR A, HENELL D, SHILKROT R, et al. Kinect@ home: crowdsourcing a large 3D dataset of real environments[C]//2012 AAAI spring symposium series. Palo Alto, CA, 2012.
12. GEIGER A, LENZ P, URTASUN R, et al. Are we ready for autonomous driving? The kitti vision benchmark suite[C]//IEEE conference on computer vision and pattern recognition. New York: IEEE, 2012: 3354–3361.
13. KOPPULA H, ANAND A, JOACHIMS T, et al. Semantic labeling of 3D point clouds for indoor scenes[J]. Advances in neural information processing systems, 2011, 24: 244–252.
14. SILBERMAN N, HOIEM D, KOHLI P, et al. Indoor segmentation and support inference from rgb-d images[C]//12th european conference on computer vision. Berlin: Springer-Verlag Berlin, 2012: 746–760.
15. SONG S, XIAO J, IEEE. Tracking revisited using rgb-d camera: unified benchmark and baselines[C]//IEEE International conference on computer vision. New York: IEEE, 2013: 233–240.
16. PARK S, CHUN M M. Different roles of the parahippocampal place area (ppa) and retrosplenial cortex (rsc) in panoramic scene perception[J]. Neuroimage, 2009, 47(4): 1747–1756.
17. HARRIS M, SENGUPTA S, OWENS J D. Parallel prefix sum (scan) with cuda[J]. GPU gems, 2007, 3(39): 851–876.
18. IZADI S, KIM D, HILLIGES O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera[C]//Proceedings of the 24th annual ACM symposium on user interface software and technology. Santa Barbara, California, USA: 2011: 559–568.