Visual Localization and Mapping with Objects in Logistics Environments

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN
(Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Eng. Benchun Zhou

Tag der mündlichen Prüfung:	21.12.2023
Hauptreferent:	Prof. Dr.-Ing. Kai Furmans
Korreferent:	Prof. Dr.-Ing. Christoph Stiller

Acknowledgement

I can't imagine that I have finished my doctoral thesis!

This is one of the most important moments in my life. The four-year overseas Ph.D. journey seems like a dream, teaching me to grow up, live, and think. As we all know, pursuing a Ph.D. could be very stressful, challenging, and struggling, but many people offered me a lot of help. Therefore, I would like to take this opportunity to thank all those who follow me, support me, encourage me, and love me.

Firstly, I would like to appreciate myself. The one who decides to venture beyond the comfort zone, explores unexpected challenges, and keeps curious through ups and downs. He was out of control, but finally, he is in control.

My sincerest gratitude goes to Prof. Dr.-Ing Kai Furmans, who opened the door to my Ph.D. life. He is a nice supervisor who provides me valuable guidance, constructive feedback, and thoughtful suggestions, which shape the direction of this thesis and enhance its quality. In addition. His ideas on academics, his experiences on projects, and his personalities play an important role in my overseas life and leave me with a lot of precious wealth.

I would like to acknowledge China and the Chinese Scholarship Council (CSC) Funding, which supports me in completing my studies. I may not be able to study abroad without it. No matter what happens, I know China is strong and always behind me, which makes me confident and motivates me to move forward.

I am deeply grateful to my colleagues for stimulating discussions, collaboration, and exchange of ideas. Andreas Trenkle, Jan-Felix Klein, Lukas Müller, Dali Sun, Constantin Enke, Lars Ohnemus, Patric Hopfgarten, Maximilian Gilles,

Jonathan Auberle, Gideon Arndt, Maximilian Ries, Market Kai, etc. Their diverse perspectives and intellectual contributions are impressive. Institute of Material Handling and Logistics is a big family that I feel at home.

I am also thankful to my friends, Da Jin, Hang Li, Meijun Zhou, Jiawen Zhang, Zhen Chen, Yan Wang, Hao Pang, Yizhuo Xi, Lihan Xie, etc., who have been with me for four years. Besides, I would like to thank all the people I met in Germany, no matter whether they were PhDs, Masters, or Bachelors, whether they were tourists who explored cities with me or hikers who enjoyed nature together. You make up my unforgettable memories.

Last but not least, I would like to express my sincere appreciation to my family, who stands behind me and supports me. Their encouragement and expectation are warm and healing. Thank my friends in China, who fork and star me, share interesting stories, and split my worries.

Standing at the turning point of my life, I feel amazing and crazy. It seems I woke up from a dream, but it is the truth. I am proud of myself! I've made it!

Karlsruhe, December 2023 Benchun Zhou

Kurzfassung

Simultaneous Localization and Mapping (SLAM) ist eines der grundlegenden Probleme in der mobilen Robotik, bei dem die Pose eines mobilen Roboters geschätzt und gleichzeitig die unbekannte Umgebung kartiert wird. Für visuelles SLAM bauen viele Methoden eine Punktkarte auf. Dies ist effizient für die Echtzeitverfolgung und die Kartenaktualisierung, aber nicht ausreichend für die Navigation oder andere interaktive Aufgaben. Die Objekterkennung und die semantische Kartierung hingegen liefern reichhaltige Informationen über die Umgebung, können die Intelligenz des Roboters verbessern und sicheres Verhalten garantieren.

In dieser Dissertation, semantische Objekte werde in visuelle Lokalisie-rungs- und Kartierungssysteme eingeführt. Die folgenden Bereiche werden von uns in den Fokus genommen: Objekterkennung, objektbasierte Lokalisierung und objektbasierte Kartierung.

Für die Objekterkennung schlagen wir drei verschiedene Methoden vor, um 3D-Objekte aus einem einzigen RGB-D-Bild zu erkennen. Bei der Sample-Score-Methode werden Hunderte von 3D-Quaderkandidaten generiert und der beste Kandidat unter physikalischen- und Bildeinschränkungen ausgewählt. Bei der Geometrie-Methode werden die Tiefeninformationen in 3D-Punkte umgewandelt und die Objektpunkte zu 3D-Quadern geclustert. Die Deep-Learning-Methode verwendet ein tiefes neuronales Netzwerk, um Wissen aus Trainingsdaten zu lernen und Objekte in unbekannten Szenen vorherzusagen. Um die Leistung der vorgeschlagenen Methoden zu vergleichen, Experimente werden durchgeführt, um ihre Erkennungsgenauigkeit und Laufzeiteffizienz zu messen. Die Sample-Score-Methode übertrifft die beiden anderen Methoden in Bezug

auf die Geschwindigkeit, aber die Deep-Learning-Methode erreicht die höchste Genauigkeit. Die Geometriemethode bietet einen Kompromiss zwischen Geschwindigkeit und Genauigkeit. Die Auswahl der Methode hängt vom Anwendungsfälle ab.

Für die objektbasierte Lokalisierung, ein Punkt-Ebene-Objekt-SLAM-System wird vorgeschlagen. Basierend auf dem state-of-the-art Punkt-SLAM-System, strukturelle Ebenen und semantische Objekte werden in jedem Bild erkannt, die Datenassoziation wird zwischen diesen Landmarken untersucht, und ein Graph wird formuliert, um Kamerapositionen und andere Komponenten zu optimieren. Darüber hinaus wird ein objektbasiertes Modul zur Loop-Erkennung entwickelt, um eine globale Lokalisierung unter großen Blickwinkeln zu erreichen. Das vorgeschlagene System ermöglicht eine Schätzung der Kameraposition und ist in der Lage, eine semantische Karte in einer unbekannten Umgebung zu erstellen. Im Vergleich zu anderen visuellen SLAM-Systemen, die dem Stand der Technik entsprechen, kann die Einführung von Objekten das Verständnis der Szene und die Lokalisierung der Kamera verbessern.

Im Zusammenhang mit dem Kartierung auf Objektebene wird eine effiziente semantische Pipeline vorgeschlagen, um die Umgebung zu rekonstruieren. Nachdem Nachdem die Objekterkennung und die Schätzung der Kamerapose gelöst wurden, werden die Objekte in ein Voxel-basiertes Kartierung-Framework integriert, um schrittweise eine globale Karte mit einzelnen Objekten zu erstellen. Anstatt rechenintensive Komponenten wie die pixelweise Segmentierung zu verwenden, nutzt die Methode 2D Bounding Boxes, um die Karte zu erstellen. Dabei weist die semantische Karte eine vergleichbare Leistung auf und vermeidet gleichzeitig hohe Rechenkosten.

Alle vorgeschlagenen Methoden werden von uns anhand von Open-Source Datensätzen für Innenräume sowie von Logistikszenarien evaluiert. Die Ergebnisse zeigen, dass die Einbeziehung von Objekten das Verständnis der Szene, die Lokalisierung und die Kartierung verbessern kann. Darüber hinaus werden Feldversuche auf einer realen Roboterplattform durchgeführt, um die Effektivität und Anwendung in der realen Welt zu demonstrieren.

Abstract

Simultaneous localization and mapping (SLAM) is one of the fundamental problems in mobile robotics, which estimates the location of a mobile robot and simultaneously maps the previously unknown environment. For visual SLAM, many methods build a feature point map, it is efficient for real-time tracking and map updating but not enough for navigation or other interactive tasks. Object detection and dense semantic mapping, on the other hand, provide rich information about environmental entities, improving a robot's intelligence and ensuring safe operation in the environment.

In this thesis, semantic objects are introduced to visual localization and mapping systems and the following fields are addressed: object detection, object-based localization, and object-level mapping.

Regarding object detection, three different methods are implemented to detect 3D objects from a single RGB-D frame. The sample-score method samples hundreds of 3D cuboid candidates and selects the best candidate based on image and physical constraints to represent the object. The geometry method involves converting depth information into 3D points and clustering object points into 3D cuboids. The deep learning method employs a deep neural network to learn knowledge from training data and is able to predict objects in unknown scenes. To compare the performances of the proposed methods, Experiments are designed to measure their detection accuracy and runtime efficiency. The sample-score method outperforms the other two methods in speed, but the deep learning method achieves the highest accuracy. The geometry method provides a compromise between speed and accuracy. Considering different use-case, different methods can be utilized.

For object-based localization, a point-plane-object SLAM system is proposed. Based on the state-of-the-art point-SLAM system, structural planes and semantic objects are detected in every key-frame, data association among these landmarks is explored, and a graph is formulated to optimize camera poses and other components. In addition, an object-based loop detection module is designed to achieve global localization under large viewpoints. The proposed system achieves camera pose estimation and is able to build a sparse semantic map in an unknown environment. Compared to other state-of-the-art visual SLAM systems, introducing objects can benefit scene understanding and improve camera localization.

In the context of object-level mapping, an efficient semantic mapping pipeline is proposed to reconstruct the environment. After solving object detection and camera pose estimation, objects are integrated into a voxel-based mapping framework to incrementally build a global map with individual objects. Rather than relying on high computational components like pixel-wise segmentation, the method capitalizes on 2D bounding boxes to build the map, which has a comparative performance on semantic mapping while avoiding high computational costs.

To demonstrate the capabilities, all proposed methods are evaluated on open-source indoor datasets, as well as logistics scenarios. The results show that the incorporation of objects can benefit scene understanding, localization, and mapping. Furthermore, field experiments are performed on a real robotic platform to demonstrate the effectiveness and application in the real world.

Contents

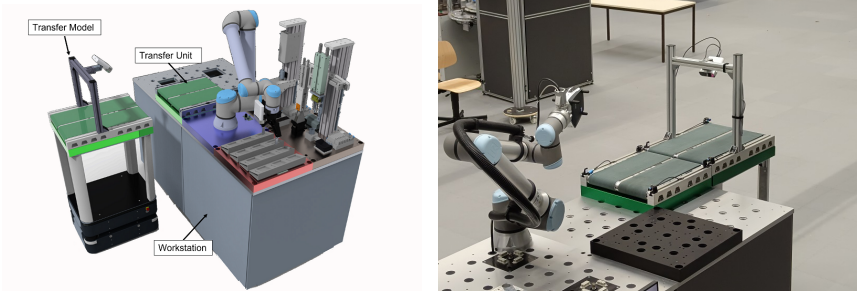
1 Introduction

The Fourth Industrial Revolution (Industry 4.0) is the trend towards automation and data exchange in manufacturing technologies and processes. It is characterized by the integration of new technologies such as cyber-physical systems, artificial intelligence, autonomous robotics, etc. Fostered by this trend, many advanced production systems are established. For example, a cyber-physical system can be found within a modular smart factory, which is able to sense the physical world, create a digital image, and make decentralized decisions. Within the context of the Internet of Things (IoT), various components have the ability to communicate efficiently and collaborate seamlessly, creating a complete supply chain. In addition, intelligent machines can be introduced to improve production efficiency without human intervention.

In the field of material handling and logistics, productivity, and reliability have increased tremendously over the last thirty years, but flexibility, robustness, and efficiency are still demanding. Furmans et al. (2018) analyzed today's material handling systems and described the desired properties of future systems, including plug-and-play capability, scalability, reconfigurability, reliability, safety, resource efficiency, etc. They also proposed suitable patterns to achieve these properties, such as modularity, decentralized control, interaction, standard physical and information interfaces, etc.

At the Robotics and Interactive Systems department (ROBiS, IFL, KIT), robots are being developed to increase flexibility in intra-logistics. This thesis focuses specifically on autonomous mobile robots, where the abilities to perceive scenarios are investigated and a semantic visual localization and mapping system is proposed.

1.1 Problem Description and Research Questions



(a) AgiProbot simulated environment.

(b) AgiProbot real environment.

Figure 1.1: The AgiProbot project. The transfer module (mobile robot) is tasked to transfer items among different workstations. Specifically, the mobile robot should recognize the transfer unit (conveyor) on the top of the workstation, build a semantic map consisting of object information, and design a collision-free path towards the destination.

Advanced industrial robots are essential components inside a manufacturing system, they can operate autonomously in the scenario and communicate directly with other entities. The AgiProbot project (Agile PROduction system using mobile, learning roBOTs with multi-sensors for uncertain product specifications) [1] (Klein et al. 2021), developed at Karlsruhe Institute of Technology (KIT), is a typical intra-logistics system for agile remanufacturing processing where the mobile robots are tasked to deliver items among different workstations. As shown in Figure 1.1, the transfer module (mobile robot) should recognize the transfer unit (conveyor) on the top of the workstation, build a semantic map consisting of object information, and design a collision-free path towards the destination. To accomplish this task, the mobile robot needs to know its location and destination, which is called the pose estimation or localization problem. Besides, to design a proper path, the mobile robot needs to map the scenario from onboard sensors,

[1] http://agiprobot.de/

which is called the mapping problem. These two problems are coupled with each other because a good localization improves mapping quality, meanwhile, a good 3D map benefits the localization. SLAM is one option that can solve them together.

Cameras and laser scanners are common sensors that can be used for indoor SLAM. Compared to other sensors, the RGB-D camera is lightweight, low-cost, and energy-efficient. Most importantly, it can provide color images and direct depth measurements, making it suitable for object detection and semantic mapping in indoor environments. Therefore, RGB-D cameras are chosen for research.

Many visual SLAM methods can build a sparse or semi-dense point cloud map, it is efficient for map updating and localization. However, this map lacks semantics and can not be used for navigation or other interactive tasks (Grinvald et al. 2019). On the other hand, object detection and dense semantic mapping can perceive the environment, allowing seamless incorporation of semantic understanding into the SLAM system. Therefore, the overall research question in this thesis can be summarized as **How to detect 3D objects from RGB-D images? Do they benefit localization and mapping?** Supported by AgiProbot project (as shown in Figure 1.1), many limitations and assumptions should be mentioned before research. For example, the mobile robot, which is responsible to deliver items from one workstation to another, is equipped with an RGB-D camera and a 2D laser scanner. However, this thesis focuses solely on utilizing the RGB-D camera for localization and mapping. Moreover, the mobile robot is controlled by an industrial computer without a GPU, thus requiring a precise and efficient system to avoid failure during real-world application. The research question is tackled by breaking it down into three parts, and solutions for each one are systematically found:

The first problem is related to 3D object detection. Objects are high-level elements in the environment and can provide several advantages, such as creating meaningful maps and benefiting scene understanding. While current research work (Jiao et al. 2019) has yielded impressive results for segmenting individual objects in images, lifting 2D detections to 3D space remains challenging. Besides, limited

computation resource is another issue when transferring this technology into application. **How to efficiently detect 3D objects from a single RGB-D frame?** Here, three methods are implemented with different sources: RGB images, depth measurements, and training data. Their detection accuracy and runtime efficiency are evaluated on indoor and logistics datasets.

The second problem is the localization of the autonomous mobile robot, namely, to keep track of a robot's location using visual information. While common approaches take feature points as landmarks to track camera poses, **can the introduction of semantic objects in the camera tracking process improve the localization accuracy.** For this purpose, a visual SLAM system is proposed, where the feature points, geometry planes, and semantic objects are integrated into a unified framework to optimize camera localization. To analyze the performance of the proposed system, experiments are designed on publicly indoor and logistics environments to compare it with other state-of-the-art SLAM systems.

The third problem focuses on semantic mapping with objects. Although SLAM methods can build a map with objects, it is sparse and can not be used for navigation. There is a lot of research work developing dense maps with RGB-D cameras, but these methods suffer from critical real-time issues due to the heavy processing components. **How to efficiently and densely map the environment with semantic object information?** Considering the real-world use case, the generated map should contain occupancy and object information in order to realize scene understanding and autonomous navigation. To achieve this goal, a CPU-based semantic mapping system is proposed, which integrates the object information into a voxel-based mapping framework to incrementally build an object-level map. The proposed system is evaluated in indoor and logistics environments to demonstrate the capabilities of dense mapping and low-cost computation.

In summary, semantic objects are investigated, and their benefits of localization and mapping are explored through experiments. Accuracy and efficiency are two important metrics when evaluated on open-source datasets. Additionally, field experiments in a real robotic platform are designed to highlight the performance of the proposed system.

1.2 Structure of the Thesis

Chapter 2 is the fundamental part of the thesis. It emphasizes the importance of object detection for scene understanding and addresses the current problems when leveraging 3D objects from 2D detection. After reviewing many feasible approaches, three different methods for 3D object detection are implemented, namely the sample-score method, geometry method, and deep learning method, which mainly depend on RGB images, depth measurements, and training data respectively. Experiments are designed to evaluate the performance of the three methods in terms of detection accuracy and runtime efficiency. Besides, the critical factors that may influence performance are also analyzed. The first question is answered.

In Chapter 3, the detected objects are introduced to a visual SLAM framework to track camera poses. State-of-the-art visual SLAM algorithms achieve impressive results with feature points, but they are likely to fail in some low-texture environments due to the lack of reliable features. In this chapter, a point-plane-object SLAM is proposed, where it can estimate more landmarks, such as geometric planes and semantic objects, design new measurement functions and data association strategies, and integrate these landmarks into a unified optimization process. The proposed method is evaluated on both indoor and logistics environments. Results show that the proposed method can localize the camera and map the environment with semantic landmarks. Furthermore, compared to other state-of-the-art visual SLAM approaches, objects can benefit scene understanding and improve localization accuracy. Therefore, the second question is answered.

Chapter 4 discusses how to build a dense map with semantic object information. After object detection and camera pose tracking, a dense semantic map can be reconstructed, which contains occupancy and object information for navigation and scene understanding. Considering real-world applications, an efficient mapping approach is desirable. In this chapter, an efficient object-level mapping system is proposed, where the object association strategy and voxel-based mapping approach are addressed. The object association strategy uses geometric and

semantic descriptors to track and update object information, and the voxel-based mapping approach is able to incrementally build a dense map of the environment. Experiments on open-source indoor and logistics environments demonstrate that the proposed method can densely build an object-level volumetric map while reducing computational costs. In this case, the third question is answered.

After responding to all questions, the final chapter summarizes the contributions of this thesis and presents an outlook on future research directions.

Here is a flowchart that shows the structure of the thesis.

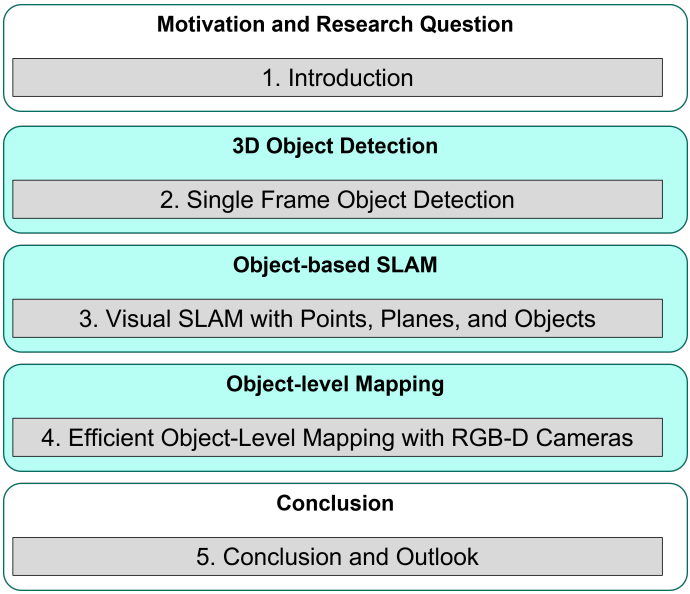


Figure 1.2: Structure of the thesis.

2 Single Frame Object Detection

In this chapter, three methods to detect 3D objects from a single RGB-D frame are proposed. Section 2.1 introduces the background of object detection, Section 2.2 reviews related work of 2D and 3D object detection, where 3D object detection methods are divided into deep learning or feature-based methods depending on whether they use neural networks or not. Section 2.3 presents three proposed methods: the sample-score method uses only RGB images, the geometry method depends mainly on depth measurements, and the deep learning method learns object attributes from the training data. Section 2.4 designs the experiments on indoor and logistics datasets to evaluate the performance of proposed methods. Finally, Section 2.5 closes the chapter. It is worth noting that this chapter only considers single frame detection, data association and multiple frame optimization are addressed in Chapter 3 and 4.

2.1 Introduction

Objects are high-level entities of the environment that carry semantic and geometric information. With the development of deep learning methods, recent research works have shown an impressive performance on object detection tasks, such as predicting object-wise bounding boxes (Redmon and Farhadi 2017) or generating pixel-wise object masks (He et al. 2017) from single RGB images. Although object detection on 2D images can be regarded as well studied (Jiao et al. 2019), 3D object perception remains a difficult challenge due to different object appearances, poses, and overlaps. Besides, real-time performance is another issue that should be considered when transferring to real-world applications.

In this chapter, 3D objects are detected from RGB-D images. For this purpose, current 3D object detection methods are reviewed on RGB-D images and three feasible approaches are implemented. The sample-score method mainly depends on the RGB image, it samples object dimension and rotation, generates candidates by camera perspective projection, and scores all candidates with image and physical constraints. The geometry method converts depth information into a point cloud, removes outliers, and clusters objects with 3D points. The deep learning method learns the attributes of the objects through training data and predicts the objects with a pre-trained neural network. To evaluate the proposed methods in terms of accuracy and efficiency, experiments are conducted on indoor and logistics datasets. Furthermore, experiments are designed to investigate the factors that affect the detection results. In summary, the contributions are as follows:

- Three different 3D object detection methods, namely the sample-score method, geometry method, and deep learning method.
- Evaluation of the proposed methods on open-source indoor datasets in terms of detection accuracy and run-time performance.
- Creation of the IFL RGB-D dataset for 3D object detection in logistics environments.
- Evaluation on the IFL RGB-D dataset to analyze the factors that may influence the detection results.

2.2 Literature Review

Object detection is an important branch of computer vision, which has been widely used in many fields of modern life, such as autonomous driving, security monitoring, and so on. The goal is to locate the object instances and assign them to a certain class within an image or video. Driven by continuous improvements of research, a series of progresses have been made to improve the object detection performance (Jiao et al. 2019). In this paper, 2D object detection is distinguished

from 3D object detection, where the 2D objects are represented as masks or bounding boxes in images, while the 3D objects are formulated as object models or other artificial geometry in 3D space. Furthermore, for 3D case, the object detection methods can be classified as deep learning methods and feature-based methods, depending on whether they use neural networks or not. Deep learning methods can achieve better detection accuracy while feature-based methods predict objects with a higher speed, some representative object detection approaches are reviewed as follows:

2.2.1 2D Object Detection

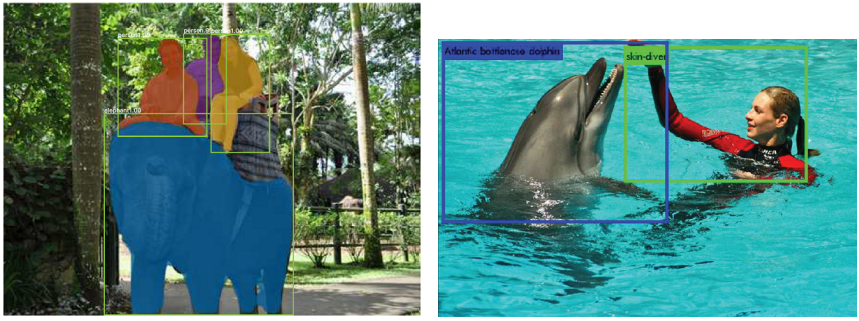
Through years of development, object detection from images has achieved good performance and can be considered a mature research field (Jiao et al. 2019). Generally speaking, current image object detectors can be divided into two categories, one is the two-stage detector, such as Mask R-CNN (He et al. 2017), and the other is the one-stage detector, such as YOLO (You Look Only Once) (Redmon and Farhadi 2017). The comparison between them is shown in Table 2.1.

The two-stage object detectors consist of two processes: region proposal and feature extraction. Take Mask R-CNN (He et al. 2017) as an example, on the first stage, it uses FPN (Feature Pyramid Network) (Lin et al. 2017) to obtain several RoI (Region of Interest) features from different levels of the feature pyramid according to their scale. On the second stage, RoIAlign is used to extract a feature map, which can be used for following classification and pixel-wise segmentation tasks. These two-stage detectors can achieve high object recognition accuracy while consuming many computation sources.

The one-stage object detectors directly predict bounding boxes from input image without region proposal step, which are time efficient and can be used for real-time devices. For instance, YOLOv2 (Redmon and Farhadi 2017) divides the input image into different grids and predicts the bounding boxes with category confidence. For each grid, many techniques can be utilized to improve the

Table 2.1: Literature review of 2D object detection.

	Name	Sensor	Application	Features
He et al. (2017)	Mask R-CNN	RGB	general	two-stage, accurate, generate pixel-wise masks
Redmon and Farhadi (2017)	YOLOv2	RGB	general	one-stage, fast, predict object-wise bounding boxes



(a) Mask R-CNN (He et al. 2017) **(b)** YOLOv2 (Redmon and Farhadi 2017)

Figure 2.1: Examples of 2D object detection methods. Mask R-CNN can generate pixel-wise masks while YOLO can predict object-wise bounding boxes. The figures come from the original work.

detecting precision and processing speed, including batch normalization, multi-scale training, fine-gain features, and so on. These one-stage detectors can reach comparative detection accuracy with a high inference speed.

2.2.2 3D Object Detection with Feature-based Methods

Although 2D object detection tasks can be regarded as well-studied, 3D object detection from a single frame is still under investigation due to limited observation and different object poses. Based on the 2D detection result, 3D object detection

methods can be divided into two different categories, one is feature-based methods, which take geometry features to leverage objects in 3D space, such as edges, colors, etc. The other is deep learning methods, they predict 3D objects directly from images by training a neural network. Some related work is summarized in Table 2.2 and 2.3.

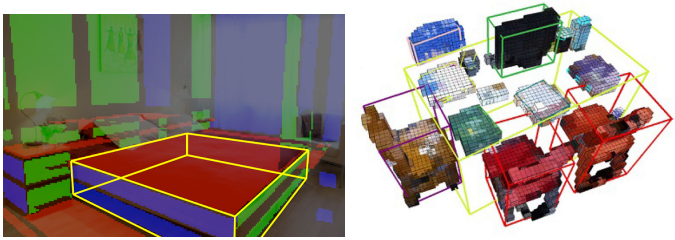
The feature-based methods are started with here. A straightforward way is to use 2D pixel-wise masks, where the 3D object points can be directly segmented from the depth image (Grinvald et al. 2019). Another simple but practical idea is to create a database to store 3D object models and match 2D detections to the database (Parkhiya et al. 2018). When prior knowledge is available in the form of 3D models (Sivananda et al. 2022), objects can be detected by matching salient key-points or features to the models. This model-based object detection method turns out to be an accurate and stable solution, but the requirement for prior 3D object models limits the application area.

Except for the pre-defined models, other artificial models, such as ellipsoids and cuboids, are more flexible and compact to represent objects with translation, rotation, and dimension. Assuming 2D detection is available, Rubino et al. (2017) proposed to initialize quadrics from multi-view frames, where several 2D bounding boxes with corresponding fitted ellipses can build a closed-form solution to initialize a 3D quadric. Nicholson et al. (2018) back-projected the edges of the 2D object bounding box to get 3D enveloping planes, thus determining an optimal dual quadric tangent to the planes. With depth measurements, Liao et al. (2020a) proposed an ellipsoid initialization from a single frame based on the relationship between objects and the supporting planes. Besides, there are a lot of works that can robustly improve the initialization of ellipsoids, such as (Liao et al. 2022), (Chen et al. 2021), and (Tian et al. 2021).

The cuboid model is another common representation of objects. Gupta et al. (2010) proposed to estimate cuboids by recovering three mutually orthogonal vanishing points. They extracted line segments, generated multiple hypotheses for rooms and objects, and selected the best configuration considering the volumetric constraints of the physical world. On this basis, Yang and Scherer (2019a)

proposed to sample cuboid corners from the 2D bounding box and generate many cuboid candidates with vanishing lines. After that, the image and physical constraints were introduced to score and select the best representation. In contrast to previous approaches that rely on detecting vanishing points of the scene and grouping line segments to form cuboids, Xiao et al. (2012) built a discriminative parts-based detector that models the appearance of the cuboid corners and internal edges while enforcing consistency to a 3D cuboid model. Besides, RGB-D images can provide additional information to initialize 3D objects because the object point cloud can be obtained. A simple idea is to formulate a cuboid that covers all corresponding points (Jiang and Xiao 2013). Besides, Mishima et al. (2019) performed a cuboid reconstruction pipeline by searching three perpendicular planes and computing the intersection of the planes. This incremental modelling framework is practical but limited to cuboid-shaped objects. For other general objects with diverse shapes, Lin et al. (2021) proposed to collect object points, remove discrete outliers, and generate the cuboid parameters in 3D space.

Due to significant variability in object appearance, overlap, and other challenges, these object detection approaches might over-segment the objects or contain noisy information, making high-level scene understanding difficult.



(a) Feature-based method using RGB information. (Gupta et al. 2010)

(b) Feature-based method using RGB-D information. (Lin et al. 2021)

Figure 2.2: Examples of 3D object detection with feature-based methods. They usually take geometry features to leverage objects in 3D space, such as edges, corners, etc. The figures come from the original work.

Table 2.2: Literature review of 3D object detection with feature-based methods.

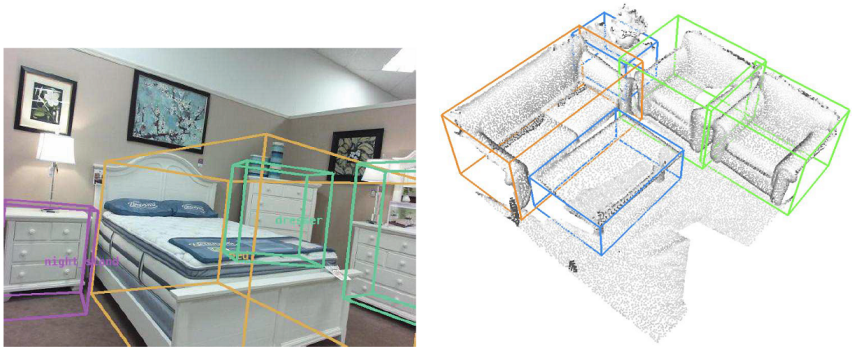
	Scene	Sensor	Framework	Model	Method
Parkhiya et al. (2018)	indoor	RGB	feature-based	points	model matching
Grinvald et al. (2019)	indoor	RGB-D	feature-based	points	mask segmentation
Rubino et al. (2017)	indoor	RGB	feature-based	quadric	2D ellipse, multi-view optimization
Nicholson et al. (2018)	indoor	RGB	feature-based	quadric	2D box to 3D enveloping planes
Liao et al. (2020b)	indoor	RGB-D	feature-based	quadric	symmetric planes
Gupta et al. (2010)	indoor	RGB	feature-based	cuboid	line segmentation
Xiao et al. (2012)	indoor	RGB	feature-based	cuboid	corner detection
Yang and Scherer (2019a)	indoor	RGB	feature-based	cuboid	vanishing lines
Jiang and Xiao (2013)	indoor	RGB-D	feature-based	cuboid	minimal cuboid
Mishima et al. (2019)	indoor	RGB-D	feature-based	cuboid	three perpendicular planes
Lin et al. (2021)	indoor	RGB-D	feature-based	cuboid	projecting onto ground

2.2.3 3D Object Detection with Deep Learning Methods

3D object instances can also be predicted directly from images with deep learning methods, they use artificial neural networks to detect and classify objects in 3D space. Given annotated training data, object features can be extracted, learned, and stored inside the network. After training, the network can be used for object detection in other unseen environments.

Focusing on the indoor environment, RGB images can provide rich features for objects, such as corners, edges, planes, etc. Dwibedi et al. (2016) proposed an end-to-end system capable of localizing cuboid vertices within an RGB image. The key idea is to first produce a 3D interpretation of the box-line object and refine the key-points by pooling convolutional features iteratively. Considering 2D-3D consistency between image and spatial space, Huang et al. (2019) proposed to predict perspective points, which are defined as the 2D projections of local Manhattan 3D key-points and satisfy geometric constraints imposed by the perspective projection. In addition to the object features, RGB images also provide the position relation between objects and other components, Huang et al. (2018) formulated a deep learning model to predict local object attributes, as well as the room layout. The individual attribute loss and physical plausibility loss penalize the intersection between the reconstructed 3D objects box and the 3D room layout, cooperatively promoting the networks. Nie et al. (2020) assumed that each object has multi-lateral relations with its surroundings, these element-wise relational features, together with object-wise appearance features are summed to predict the 3D bounding boxes of each object.

3D objects can also be estimated from depth images or 3D point clouds. For example, PointNet (Qi et al. 2017a) a pioneer work in this direction, the key idea is to use the network to learn a set of optimization functions/criteria that select interesting or informative points of the point cloud and aggregate these learned optimal values into the classification or segmentation. In the following, PointNet ++ (Qi et al. 2017b) introduces a hierarchical neural network to recursively capture geometric structures from small neighbourhoods, extending the ability to recognize fine-grained patterns and generalizability to complex scenes. VoteNet



(a) Total 3D understanding. (Nie et al. 2020)

(b) VoteNet. (Qi et al. 2019)

Figure 2.3: Examples of 3D object detection with deep learning methods. They usually use artificial neural networks to detect and classify objects in 3D space. The figures come from the original work.

(Qi et al. 2019) is proposed to generate high-quality object proposals by learning to vote object centroid directly from point clouds and aggregate votes through their features and local geometry. On this basis, ImVoteNet (Qi et al. 2020) designs a joint 2D-3D voting scheme to leverage both geometric and semantic cues in 2D images, achieving the state-of-art 3D object detection performance on indoor datasets.

To sum up, object detection methods with deep learning neural networks can achieve significant detection accuracy. However, they also have two disadvantages. One is the requirement of large amounts of training data for robust and accurate detection, and the other is the high computational demand. Currently, these methods are time-consuming and cannot be applied to mobile robotic platforms.

Table 2.3: Literature review of 3D object detection with deep learning methods.

	Scene	Sensor	Framework	Model	Method
Dwibedi et al. (2016)	indoor	RGB	network	cuboid	corner regression
Huang et al. (2019)	indoor	RGB	network	cuboid	perspective points
Huang et al. (2018)	indoor	RGB	network	cuboid	size, rotation, distance regression, cooperative training
Nie et al. (2020)	indoor	RGB	network	cuboid	relationship with surrounding
Qi et al. (2017a)	indoor	RGB-D	network	points	interesting points selection
Qi et al. (2017b)	indoor	RGB-D	network	points	geometric structure with neighbourhoods
Qi et al. (2019)	indoor	RGB-D	network	points	object center voting and geometry constraints
Qi et al. (2020)	indoor	RGB-D	network	points	2D-3D consistency, voting schedule

2.3 Methods

In the thesis, objects are represented as 3D cuboids with geometric and semantic descriptors, including class categories, 2D bounding box, associated 3D points, cuboid parameters, and so on. The mathematic description for 3D cuboids consists of 9 DoF (degrees of freedom) parameters: 3 DoF translation $\mathbf{T} = (t_x, t_y, t_z)^\top$, 3 DoF rotation $\mathbf{R} = (\theta_x, \theta_y, \theta_z)^\top$, and 3 DoF dimension $\mathbf{D} = (d_x, d_y, d_z)^\top$. The cuboid coordinate frame is located at the cuboid center, aligned with the main axes.

Given a single frame, three different methods are implemented to estimate objects with the cuboid representation in 3D space, namely the sample-score method,

geometry method, and deep learning method. Since 2D object detection can be regarded as well-studied (Jiao et al. 2019), it is assumed that the 2D bounding boxes are available, and the focus is on lifting 2D detections to 3D cuboids. Each method is described as follows:

2.3.1 Method 1: Sample-Score Method

Section 2.3.1 is based on (Zhou et al. 2022). Parts of the following text are taken from that publication without changes.

Taking RGB images as input, the sample-score method employs a 2D object detector to get object classes and 2D bounding boxes. For each bounding box, it samples the dimension and rotation of the object to generate many cuboid candidates. These candidates are scored by 3D and 2D constraints, and the best candidate with the highest score is selected as the representation of the object. The whole process is shown in Figure 2.4.

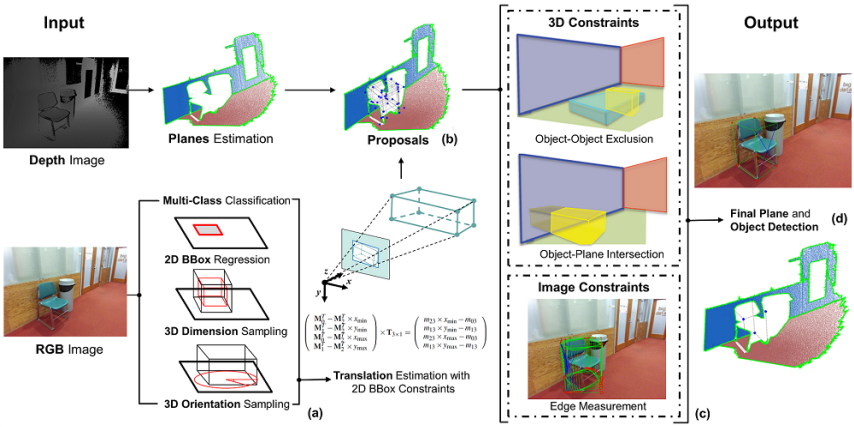
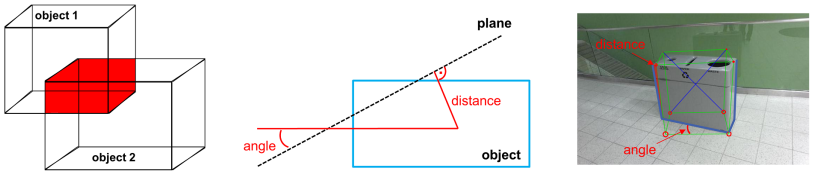


Figure 2.4: Overview of the sample-score method on 3D object detection. The method takes a single frame as input, samples object dimension and rotation to generate cuboid candidates (**a**), and scores these candidates (**b**) with spatial and image constraints (**c**). The best candidate is selected to represent the object (**d**). More details can be found in (Zhou et al. 2022).

1) Sample Process. It is possible to sample object translation, rotation, and dimension from 0 to ∞, but not practical. Given object bounding boxes in 2D images and perspective projection, the method samples the rotation and dimension and lifts the objects to 3D space. Under the assumption that objects are parallel to the ground, only one rotation angle related to the ground needs to be sampled. Besides, object dimensions can be sampled based on the average value of categories. Therefore, the sampling space is greatly reduced and becomes more accurate. Each sampled dimension and rotation, together with a 2D bounding box, can be regarded as a candidate.

2) Lifting 2D to 3D. According to the perspective projection principle: the projected vertexes of a 3D cuboid should fit tightly into each side of its 2D detection box. That means, four out of eight vertexes of the 3D bounding box should be projected right on the four sides of the 2D bounding box (Mousavian et al. 2017). According to this 2D-3D fitting constraints, once the 2D bounding box, dimension, and rotation are available, the translation of the candidate can be computed, and 3D cuboid of the candidate can be leveraged, more details can be found in Appendix A. To this end, a set of candidates can be obtained, and each candidate is formulated as 3D cuboids with 9 DoF parameters.

3) Scoring Process. Given a set of cuboid candidates, the scoring process aims to select the best proposal that not only satisfies the spatial constraints of the physical world, but also matches the local surface geometry estimated from image cues.



(a) Object-Object Constraints: 3D IoU (*front view*)
(b) Object-Plane Constraints: angle & distance (*top view*)
(c) Object-Image Constraints: angle & distance (*projected view*)

Figure 2.5: Proposed constraints for object scoring. (**a**) Object-object constraints are calculated by object 3D overlapping. (**b**) Object-plane constraints are calculated with angle and distance errors. (**c**) Object-image constraints are calculated by the angle and distance error of the image lines and projected cuboid edges.

To achieve this goal, spatial and image constraints are proposed for scoring, as illustrated in Figure 2.5, The spatial constraints (Gupta et al. 2010) can be defined with three rules: object occupancy, object-object exclusion, and object-plane constraints. They can be explained as: (1) Every cuboid candidate in 3D space should have a non-zero finite volume; (2) The volumes occupied by different objects are mutually exclusive; (3) Every object should be close to its associated planes without intersection. When projecting the 3D cuboid to the image, the image constraints are defined as the alignment between the projected cuboid edges and detected lines in the image. In summary, the overall cost can be calculated as:

$$E(O, \Pi, I) = k_1 \times \Psi_{O-O} + k_2 \times \Psi_{O-\Pi} + k_3 \times \Psi_{O-I}, \tag{2.1}$$

where Ψ_{O-O}, $\Psi_{O-\Pi}$, Ψ_{O-I} represent object-object, object-plane, and object-image constraints respectively, k_1, k_2 and k_3 are coefficients. Based on equation (2.1), the cuboid candidate with the smallest cost $E(O, \Pi, I)$ is selected as the object proposal.

2.3.2 Method 2: Geometry Method

Section 2.3.2 is based on (Zhou et al. 2023). Parts of the following text are taken from that publication without changes.

The geometry method mainly focuses on depth information and aims to cluster the object with 3D points. The whole process can be divided into three steps: Firstly, given an RGB-D frame with 2D bounding boxes, the 3D points inside each bounding box can be obtained. Since there might be outliers, the next step is to remove these outliers and cluster object points. Finally, object parameters are computed, and a cuboid is drawn to cover all points. Figure 2.6 describes the whole process.

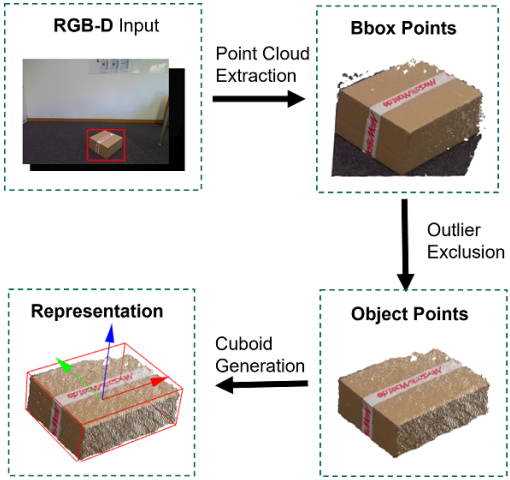


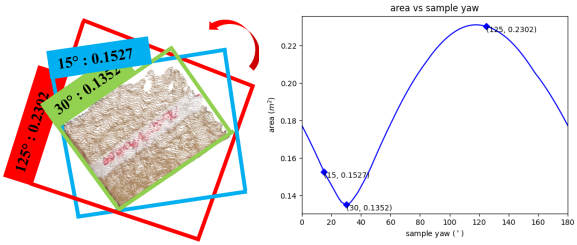
Figure 2.6: Overview of the geometry method on 3D object detection. Given an RGB-D frame and 2D bounding box, the 3D points inside each bounding box can be obtained, then, an outlier exclusion algorithm is adopted to cluster object points, finally, the cuboid parameters are computed.

1) Point Cloud Extraction. For each RGB-D frame, the geometry method first chooses YOLO (Redmon and Farhadi 2017) network to detect the object classes together with 2D bounding boxes, which is comparatively accurate and requires less computation cost. Next, the ground should be removed. To do so, each RGB-D frame is converted into a 3D point cloud and a multiple plane estimation method (Trevor et al. 2013) is applied to detect all planes in the frame. For each plane, if its normal is parallel to the world coordinate and its relative distance is zero, it can be assumed as ground and the ground points can be removed. Therefore, once the object bounding box is available in the image, the corresponding point cloud without ground points can be retrieved.

2) Outlier Exclusion. Due to the presence of sensor noise and detection errors, there are many discrete outliers in the preliminary object model. In this case, a statistical outlier removal step (Rusu and Cousins 2011) is performed to increase the accuracy of the point cloud. Additionally, a Euclidean cluster extraction (Rusu and Cousins 2011) is implemented to separate points into several groups

and cluster object points. To ensure reliability, only one cluster is selected, which should encompass at least 70% of all points.

3) Cuboid Generation. After obtaining object points, the cuboid model with 9 DoF parameters can be computed. To achieve this goal, the method first transforms all point clouds into the world coordinate. Building on the assumption that real-world objects are parallel to the ground, object roll θ_x and pitch θ_y are 0, The translation t_z and the height d_z should be calculated as the mean and difference of the maximum and minimum value of the z axis among all points. Finally, the remaining $\{t_x, t_y, d_x, d_y, \theta_z\}$ can be determined by projecting all object points on the 2D X-Y plane. As illustrated in Figure 2.7, the method performs a discrete sampling of rectangle rotation from $0°$ to $180°$ and draws oriented rectangles to cover all projected points. For all sampled oriented rectangles, the smallest rectangle matches the object points best, whose parameters (the position, dimension, and yaw angle $\{t_x, t_y, d_x, d_y, \theta_z\}$) can be adopted to represent the 3D objects.



(a) Object yaw sampling. **(b)** Areas under different object yaws.

Figure 2.7: Proposed object yaw computation in Figure 2.6. (**a**) Object yaw sampling process, (**b**) The area of oriented rectangles under different yaw angles.

2.3.3 Method 3: Deep Learning Method

Inspired by the neural network, a deep learning method is also developed to detect 3D objects from RGB images. The framework is illustrated in Figure 2.8, where a

two-stage network is proposed to learn object attributes, estimate the object class, object pose, size, etc., and recover cuboid details in 3D space.

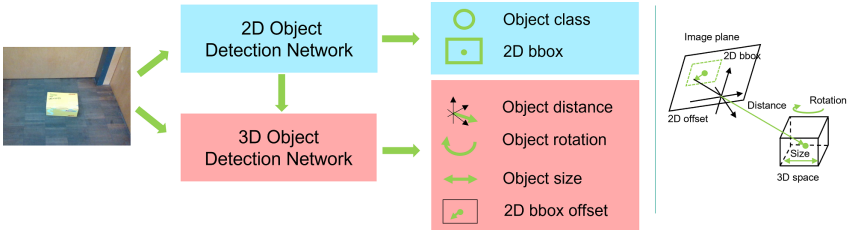


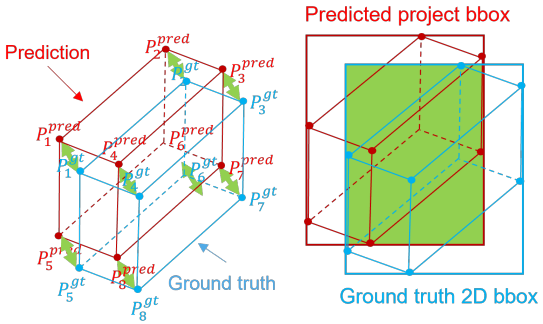
Figure 2.8: Overview of the deep learning method on 3D object detection. A two-stage neural network is proposed to learn object attributes, including object class, relative distance, pose, etc., which can parameterize a 3D bounding box.

1) Network Architecture. The 2D object detection network aims to estimate object classes and 2D bounding boxes and provide several crop images for the 3D object detection network. For 3D parameters, under the assumption that all objects are parallel to the ground, the object rotation, size, and distance from the camera center to the object center can be easily learned. However, estimating the 3D object center directly from the 2D image may result in a large variance (Huang et al. 2018), so the network bridges the gap between 2D and 3D object bounding boxes by an offset between the 2D bounding box center and the projection of the 3D bounding box center. In other words, given the camera intrinsic $\mathbf{K}$. rotation $\mathbf{R}$, translation $\mathbf{T}$, The projection relationship between the 3D center $\mathbf{C_{3d}}$, 2D center $\mathbf{C_{2d}}$, and 2D offset $\delta_{\mathbf{2d}}$ is expressed as:

$$\mathbf{C_{2d}} + \delta_{\mathbf{2d}} = \mathbf{K}[\mathbf{R}|\mathbf{T}] * \mathbf{C_{3d}} \tag{2.2}$$

In this way, the 3D and 2D parameters are united, which helps maintain the 2D-3D consistency and reduces the variance of the 3D bounding box estimation.

2) Loss Functions. The loss function is used to measure the difference between the predicted output and the ground truth. In the case, the individual loss is distinguished from the joint loss functions for the output. The object class loss



(a) 3D loss. **(b)** 2D loss.

Figure 2.9: Proposed joint loss functions for the deep learning method. (**a**)The 3D loss is calculated by the 3D distance of corners in 3D space. (**b**)The 2D loss is calculated by the 2D IoU of the projected bounding box. The red one represents the predicted cuboid, while the blue one is the ground truth.

uses cross-entropy, which separates all categories as multiple bins and measures the probability distribution in each bin. A similar idea is used for distance and rotation prediction, where the range is divided into multiple bins and the distribution in each bin is estimated. Other outputs, such as 2D bounding box, 2D offset, and size, use squared error to regress the value. Except for individual loss functions, 3D and 2D joint loss functions are also designed to cooperatively optimize the network. Ideally, the corners of the prediction and the ground truth should match, so, the 3D corner loss is defined as the corner position difference as shown in equation (2.3). On the image plane, the projected bounding box should also match the 2D bounding boxes, and 2D intersection over union is defined as the 2D loss, as shown in equation (2.4). To summarise, the total loss can be written as shown in equation (2.5), where k_1 and k_2 are coefficients to balance the different losses.

$$L_{3d} = \frac{1}{8}\sum_{i=1}^{8} ||X_{pred\_3d} - X_{gt\_3d}||^2 \tag{2.3}$$

$$L_{2d} = \Delta x_{2d} \times \Delta y_{2d} \tag{2.4}$$

$$L_{total} = (L_{class}+L_{bbox}+L_d+L_\theta+L_{size}+L_{offset})+k_1*(L_{3d}+k_2*L_{2d}) \quad (2.5)$$

2.4 Experiments

To evaluate the performance of the proposed methods, experiments are conducted on a publicly available object detection dataset, called SUN RGB-D dataset (Song et al. 2015). This dataset comprises $10,335$ images captured by 4 different RGB-D cameras, featuring over 800 object categories and $64,595$ annotated 3D bounding boxes. With such extensive data, this dataset serves as an ideal evaluation platform for 3D object detection.

Objects from logistics environments are also planned to be detected, but there are no open-source datasets for this goal. Besides, the images in the SUN RGB-D dataset are captured randomly, making it difficult to analyze the factors that influence the detection accuracy. To solve the above problems, there is an interest in generating a logistics dataset for 3D object detection, named IFL RGB-D dataset. To this end, typical logistics items are captured, such as delivery boxes, pallets, and containers, and some potential factors are addressed, including different backgrounds, categories, relative distances, and rotations. After annotation, the object detection methods are evaluated on the created IFL RGB-D dataset.

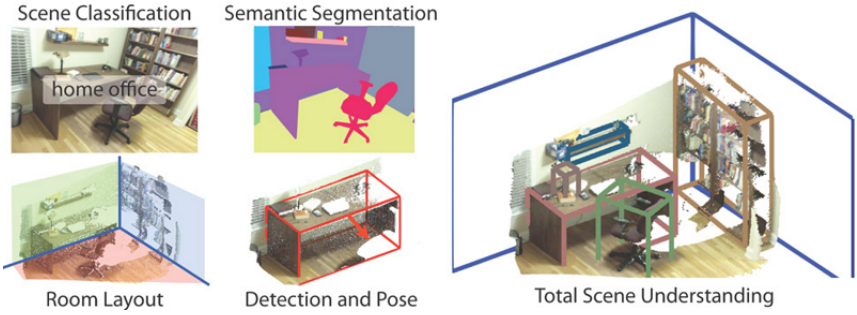


Figure 2.10: SUN RGB-D dataset. This dataset provides diverse objects and bounding box annotation. The figure comes from the official website: https://3dvision.princeton.edu/projects/2015/SUNrgbd/.

2.4.1 Experiments on Indoor Datasets

Section 2.4.1 is based on (Zhou et al. 2022). Parts of the following text are taken from that publication without changes.

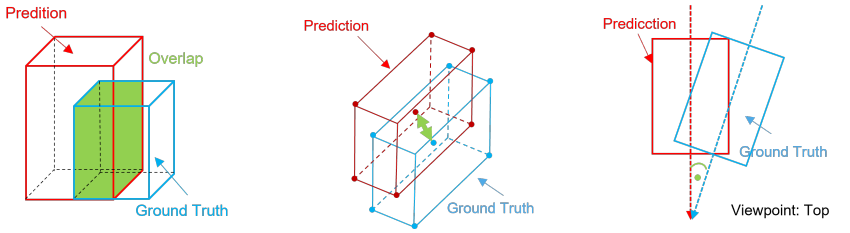
SUN RGB-D dataset (Song et al. 2015) is an indoor dataset that focuses on 6 important recognition tasks towards total scene understanding, including object detection, scene recognition, and room layout estimation. this dataset is selected because of the diverse objects and bounding box annotations. In the experiments, 8 common objects out of 1670 images are selected for further evaluation, they are bed, $sofa$, $chair$, $sofa\_chair$, $garbage\_bin$, $night\_stand$, $lamp$, and $table$ respectively.

To focus on 3D object detection, the 2D bounding box and class name are adopted from the ground truth. The sample-score method samples the object dimension around the average value between 80% and 120% with a sample step of 20% to avoid invalid parameters. For rotation sampling, it is sufficient to only sample the yaw angle from $0°$ to $180°$ with a sample step of $5°$. That means it can generate $3 \times 3 \times 3 \times 36 = 972$ cuboid candidates for each object. The deep learning method trains the neural network on 2000 images with 100 epochs and uses the pre-trained weights to predict objects in the unseen background.

Under the assumption that all objects are parallel to the ground, results are measured with 4 metrics: 3D object Intersection over Union (IoU_{3D}), per-class Average Precision, 3D translation error (E_{trans}), and rotation error (E_{rot}), which are illustrated in Figure 2.11 and defined as follows:

3D Object IoU (IoU_{3D}): the object IoU shows the general similarity of two objects, which is calculated as the intersection by the union of the 3D bounding box between the detected object and the ground truth, as shown in equation (2.6) and Figure 2.11(**a**).

per-class Average Precision (AP): the AP measures the ability to correctly detect objects in the scene. Regarding 3D object detection, cuboids are considered positive detections (TP) if the 3D IoU is greater than 25%; otherwise, they are



(a) 3D object IoU **(b)** 3D translation error **(c)** 3D rotation error

Figure 2.11: Example of evaluation metrics. (**a**) 3D object IoU is calculated as the intersection by the union of the 3D bounding boxes. (**b**) 3D translation error is measured by the Euclidean distance of the 3D cuboid centroid. (**c**) 3D rotation error is computed by the yaw angle of the 3D bounding boxes in the top view.

classified as false detections (FP). The AP is equal to the number of positive detections divided by all detections, as shown in equation (2.7).

3D Translation Error (E_{trans}): the translation error calculates the difference between the predicted centroid position and its true position of an object in 3D space, the Euclidean distance of the cuboid centroid is used to represent the error, as shown in equation (2.8) and Figure 2.11(**b**).

3D Rotation Error(E_{rot}): under the assumption that all objects are parallel to the ground, the rotation error is computed by the heading angle or yaw angle error between the detection objects and the ground truth in the top view, as shown in equation (2.9) and Figure 2.11(**c**).

$$IoU_{3D} = \frac{V_{\text{overlap}}}{V_{gt} + V_{\text{pred}} - V_{\text{overlap}}} \tag{2.6}$$

$$AP = \frac{1}{n} \sum_{i=1}^{n} \frac{|TP_i|}{|TP_i| + |FP_i|} \tag{2.7}$$

$$E_{\text{trans}} = \sqrt{|t_{x\_pred} - t_{x\_gt}|^2 + |t_{y\_pred} - t_{y\_gt}|^2 + |t_{z\_pred} - t_{z\_gt}|^2} \tag{2.8}$$

$$E_{\text{rot}} = |\theta_{z\_pred} - \theta_{z\_gt}| \tag{2.9}$$

1) Qualitative Results. A subset of the object detection results of each method are visualized in Figure 2.12, Figure 2.13, and Figure 2.14. The RGB images with 2D bounding boxes are shown in the first column, which acts as the input of the proposed system, while the last two columns show the object detection results in image and 3D space.

The sample-score method depends on edge features and planes to score the proposals, thus works best for "boxy with clear edge" objects near the wall. The geometry method relies on the quality of points and may cluster only part of the points due to different viewpoints and sensor noise. The deep learning method can predict a good cuboid by maintaining 2D-3D consistency.

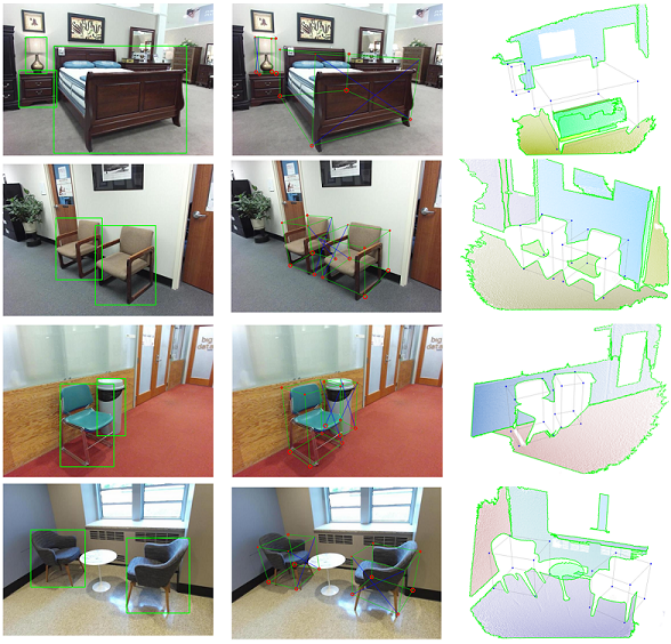


Figure 2.12: Example of detection results using the sample-score method on SUN RGB-D dataset. The RGB images with 2D bounding boxes are shown in the first column, while the last two columns show the object detection results in image and 3D space.

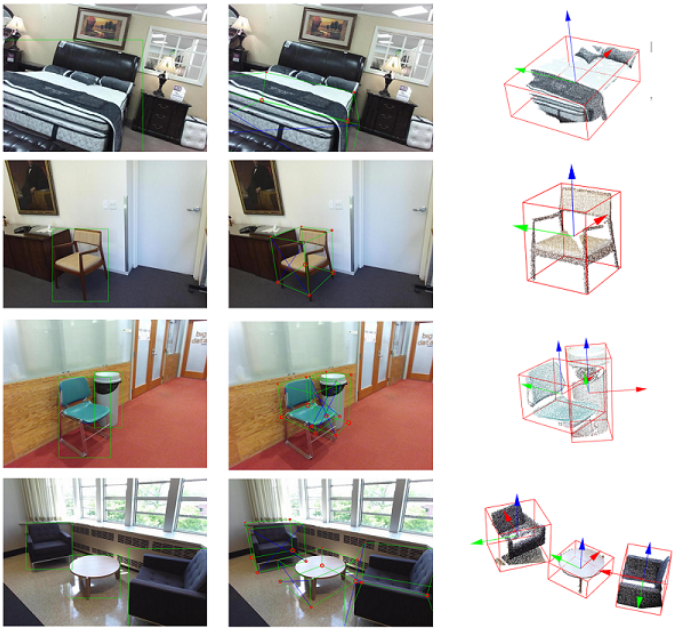


Figure 2.13: Example of detection results using the geometry method on SUN RGB-D dataset. The RGB images with 2D bounding boxes are shown in the first column, while the last two columns show the object detection results in image and 3D space.

Table 2.4: Evaluation of 3D Object IoU (IoU_{3D}) on SUN RGB-D dataset.

Method	*bed*	*sofa*	*sofa_ chair*	*chair*	*garbage bin*	*night_ stand*	*table*	*lamp*	average
count	189	40	86	180	112	160	86	186	
Sample-score method	**0.4979**	0.3189	0.3740	0.3988	0.3786	0.3559	0.2578	0.2576	0.3549
Geometry method	0.4105	**0.4258**	**0.4580**	0.3523	0.3805	0.2971	0.2344	0.1653	0.3404
Deep learning method	0.4271	0.4193	0.3775	**0.4559**	**0.4453**	**0.4029**	**0.4559**	**0.3821**	**0.4207**

Figure 2.14: Example of detection results using the deep learning method on SUN RGB-D dataset. The RGB images with 2D bounding boxes are shown in the first column, while the last two columns show the object detection results in image and 3D space.

Table 2.5: Evaluation of per-class AP on SUN RGB-D dataset.

Method	*bed*	*sofa*	*sofa_chair*	*chair*	*garbage bin*	*night_stand*	*table*	*lamp*	average
count	189	40	86	180	112	160	86	186	
Sample-score method	**0.7371**	0.5319	0.6162	0.6613	0.5714	0.4687	0.3181	0.2435	0.5189
Geometry method	0.5215	**0.6808**	**0.6744**	0.6508	**0.6837**	0.2937	0.3181	0.1474	0.4963
Deep learning method	0.6546	0.5758	0.5882	**0.7107**	0.6275	**0.6000**	**0.6552**	**0.5714**	**0.6229**

Table 2.6: Evaluation of 3D translation error E_{trans} on SUN RGB-D dataset.

Method	bed	$sofa$	$sofa\_chair$	$chair$	$garbage\_bin$	$night\_stand$	$table$	$lamp$	average
count	189	40	86	180	112	160	86	186	
Sample-score method	**0.3971**	0.4352	0.3244	0.1804	0.1633	0.2420	0.5165	0.2376	0.3120
Geometry method	0.6176	**0.2479**	**0.2542**	0.1913	**0.1331**	0.2542	0.3476	0.2225	**0.2835**
Deep learning method	0.6426	0.4961	0.3435	**0.1597**	0.2379	**0.1993**	**0.2926**	**0.1739**	0.3182

Table 2.7: Evaluation of 3D rotation error E_{rot} on SUN RGB-D dataset.

Method	bed	$sofa$	$sofa\_chair$	$chair$	$garbage\_bin$	$night\_stand$	$table$	$lamp$	average
count	189	40	86	180	112	160	86	186	
Sample-score method	9.1	9.0	9.1	**9.4**	**6.2**	9.4	**7.8**	**10.3**	8.78
Geometry method	**5.8**	**7.3**	**7.0**	11.0	9.3	**8.1**	10.1	10.4	**8.62**
Deep learning method	8.4	8.8	7.7	9.5	7.0	13.5	12.4	10.6	9.73

2) Quantitative Results. A quantitative analysis of the proposed methods regarding the 3D IoU, per-class AP, translation error, and rotation error for different object categories can be found in Table 2.4, Table 2.5, Table 2.6, and Table 2.7. When considering 3D IoU and per-class AP, the deep learning method outperforms the other two in five categories, showing a better and more robust result. Besides, it can predict small objects such as lamps, where the detection accuracy is twice as high as that of the other methods. However, the deep learning method does not perform well in translation and rotation estimation. In terms of translation, the geometry method is stable because it calculates the centroid of the point cloud. For rotation, both the sample-score method and geometry method perform better because they employ discrete object rotation samples and use geometric constraints to select the best one.

Take the geometry method as an example to analyze the performance of different object categories: bed performs best with the highest 3D IoU, This can be explained by the ease with which its large 2D bounding box and corresponding surface planes can be detected. $lamp$ is quite small, making it hard to distinguish from the background, and thus having the worst detection accuracy. Besides, $table$ and $chair$ do not have the expected result, because sometimes the feet are missing when the cluster method is used.

3) Runtime Performance. Except for the detection metrics, the runtime performances of all methods are also evaluated on a ThinkPad machine.[1] (i7-8565U 1.80 GHz CPU, 16 GB RAM, no GPU, Ubuntu 18.04). The sample-score method and geometry method are programmed with C++, while the deep learning method uses Python. The results are listed in Table 2.8. The sample-score method, which involves line segmentation, plane estimation, sampling, and scoring, utilizes mainly RGB information, thus requiring less time. The geometry method, consisting of point cloud processing, outlier exclusion, and clustering, shows a relatively slower speed. Additionally, the deep learning method operates the slowest on the CPU.

[1] The deep learning method is trained on another GPU but predicts objects using a CPU-only platform.

Table 2.8: Evaluation of runtime performance on SUN RGB-D dataset.

Method	Main tasks	Average time (mSec) *
Sample-score method	line segmentation, plane estimation, sampling, and scoring	**53.07**
Geometry method	point cloud converting, outlier exclusion, clustering	57.04
Deep learning method	network loading, predicting	157

* The 2D object detection on RGB images is ignored, which requires 725ms in the test machine.

Table 2.9: Comparison of three proposed methods on SUN RGB-D dataset.

Method	3D IoU	AP	Translation error (m)	Rotation error (°)	Runtime (mSec)
Sample-score method	0.3549	0.5189	0.3120	8.78	**53.07**
Geometry method	0.3404	0.4963	**0.2835**	**8.62**	57.04
Deep learning method	**0.4207**	**0.6229**	0.3182	9.73	157

4) Summary. The performances of the evaluated methods are summarized as shown in Table 2.9, and they have their own pro and cons. For example, the deep learning method achieves the best performance on 3D IoU and per-class AP, whereas the geometry method shows the best results on translation and rotation error. Of the three methods, the sample-score method is the fastest, while the deep learning method achieves the highest accuracy. The geometry method offers intermediate performance in both speed and accuracy.

2.4.2 Experiments on Logistics Datasets

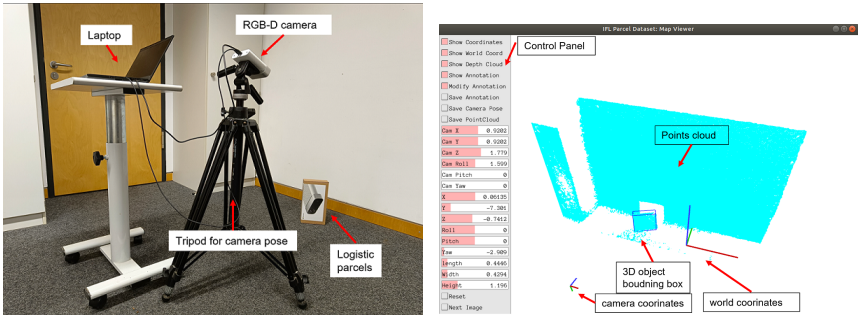
The experiments on the SUN RGB-D dataset have verified that the evaluated methods can detect 3D objects in indoor environments, but how about logistics objects, do they have the same accuracy and efficiency? To answer this question, the implemented methods are evaluated in logistics environments.

LOCO (Logistics Objects in Context) dataset (Mayershofer et al. 2020) is a logistics dataset for scene understanding, it contains 5,593 annotated images and in total 151,428 instances of pallets, small load carriers, stillages, forklifts, and pallet trucks are annotated. However, this dataset focuses on image detection and does not provide 3D information, which is not suitable for the goal. Since there is no open-source dataset available for 3D object detection in logistics environments, an IFL RGB-D dataset can be created.

Creation of the IFL RGB-D Dataset. Considering further application, two RGB-D cameras are chosen in the institute, a Microsoft Azure Kinect camera [2], and a Realsense D435 camera [3]. Calibration is done with a ROS system [4]. Figure 2.15(**a**) illustrates the capture system, where the camera is mounted on a flexible tripod that allows for convenient camera adjustment.

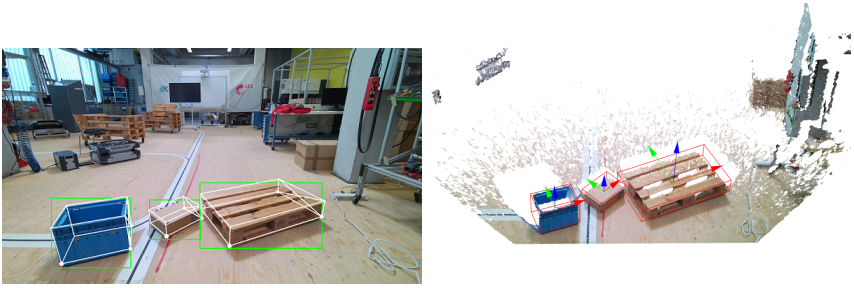
After gathering RGB-D images, an annotation tool is developed to label the ground truth, as shown in Figure 2.15(**b**). The origin of the world coordinates is located in the center of the camera, while the main axis is parallel to the ground. To align the world coordinates to the ground, this tool can identify the ground plane and rotate the coordinate system in the control panel. Then, it can search all points that belong to the object and draw a fake 3D cuboid to cover them. The 3D cuboid is also projected onto the image plane to obtain the 2D bounding box. Finally, it can save the camera setting and the object representation in the world coordinates, including the class category, 2D bounding box, translation, rotation, and dimension.

[2] https://azure.microsoft.com/en-us/products/kinect-dk
[3] https://www.intelrealsense.com/depth-camera-d435/
[4] http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration



(a) Capture system **(b)** Annotation tool

Figure 2.15: Creation of the IFL RGB-D dataset. (**a**) A capture system is designed to collect RGB-D images. (**b**) An annotation tool is developed to label the ground truth.



(a) The ground truth is plotted on the image plane. **(b)** The ground truth is plotted in 3D space.

Figure 2.16: IFL RGB-D dataset. This dataset aims to detect 3D logistics objects and analyze the factors that may influence the detection result.

While other datasets randomly capture objects, the IFL RGB-D dataset is designed specifically to analyze the factors that may influence detection accuracy, including background, categories, relative distance, and relative object rotation. To investigate these factors, a subset of objects are recorded under these different conditions during the capturing process. In total, the dataset is recorded in 6 different logistics scenes, where 3 different objects (parcel, pallet, load carrier) are emphasized, and in total 800 labels are manually annotated. One example of the dataset is shown in Figure 2.16.

Evaluation on IFL RGB-D Dataset. Among different methods, the performances of the geometry method are evaluated (see Section 2.3.2) on the IFL RGB-D dataset. Similar to Section 2.4.1, 3D object IoU, per-class AP, 3D translation error, and 3D rotation error are adopted as the evaluation metrics.

Regarding qualitative results, a subset of detection results with different backgrounds and object categories are visualized in Figure 2.17, where the backgrounds are named from $lab1$ to $lab4$, and the detected cuboids are projected onto the image plane. The results show that the proposed geometry method can also estimate 3D cuboids under different backgrounds.

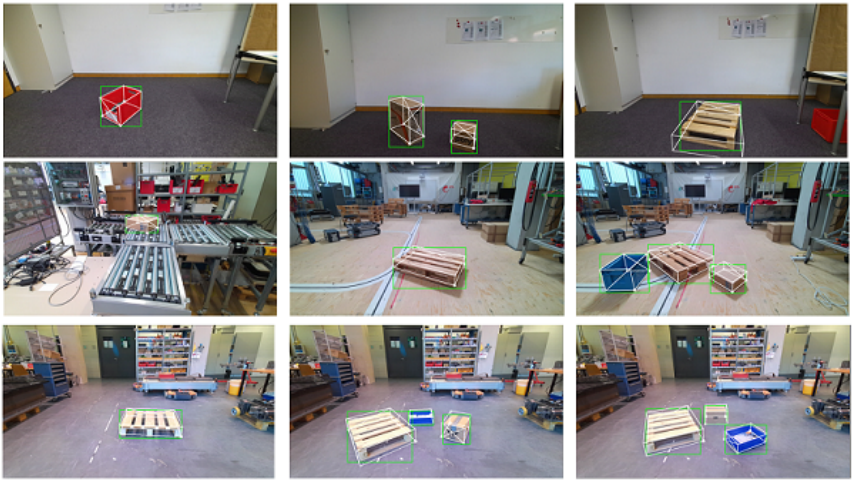


Figure 2.17: Example of detection results on IFL RGB-D dataset, where the backgrounds are named from "$lab1$" to $lab4$.

In the context of different impact factors, different experiments are designed. Firstly the experiments evaluate different class categories in background $lab1$, the results are reported in Table 2.10. The $parcel$ performs the best due to its cuboid shape, while the $pallet$ and the $load\ carrier$ are more difficult because of the small height and the absence of a top surface. Then, $parcel$ is chosen to evaluate the influence of background. As shown in Table 2.11, the geometry method is

stable under different backgrounds. Finally, $parcel$ is placed in $lab1$ with different object rotations and distances, the results are shown in Table 2.12, and Table 2.13. The detection accuracy varies. For relative object rotation, the best performance can be found on $6 * \pi/8$ because three object surfaces can be observed from this point of view. For different distances, the geometry method works best within a range of $0 - 3m$, because of the measurement limitation. These two factors are valuable because bad detection can be figured out and removed in real world applications.

Table 2.10: Evaluation of the geometry method's performance under different categories on IFL RGB-D dataset.

	$parcel$	$pallet$	$loadcarrier$
count	40	40	40
IoU_{3D}	0.6725	0.5782	0.6023
AP	0.8375	0.7455	0.6210
$E_{trans}(m)$	0.0958	0.2322	0.1156
$E_{rot}(^\circ)$	3.2	4.7	3.5

Table 2.11: Evaluation of the geometry method's performance under different backgrounds on IFL RGB-D dataset.

	$lab1$	$lab2$	$lab3$	$lab4$
count	40	50	45	40
IoU_{3D}	0.4459	0.4038	0.4295	0.4208
AP	0.8646	0.7925	0.8314	0.8708
$E_{trans}(m)$	0.1167	0.1379	0.0817	0.1020
$E_{rot}(^\circ)$	4.2	3.7	2.9	2.5

Table 2.12: Evaluation of the geometry method's performance under different yaws on IFL RGB-D dataset.

	0	$1\pi/8$	$2\pi/8$	$3\pi/8$	$4\pi/8$	$5\pi/8$	$6\pi/8$	$7\pi/8$
count	96	76	52	40	48	64	88	87
IoU_{3D}	0.4469	0.4303	0.4444	0.3818	0.4088	0.4537	**0.4782**	0.4171
AP	0.8646	0.8684	0.8462	0.7752	0.7292	0.8750	**0.8864**	0.7442
$E_{trans}(m)$	0.1161	0.1379	0.1587	0.1485	0.1470	0.1539	**0.1135**	0.1770
$E_{rot}(^\circ)$	4.2	6.3	7.9	4.6	4.4	4.8	3.8	**3.5**

Table 2.13: Evaluation of the geometry method's performance under different distances on IFL RGB-D dataset.

	0.5m	1m	1.5m	2m	2.5m	3m	3.5m	4m	4.5m
count	9	147	252	32	1	9	41	23	14
IoU_{3D}	0.4890	0.5144	0.4083	**0.6276**	0.4854	0.3353	0.4207	0.3724	0.3268
AP	1	0.8163	0.7494	0.8625	**1**	**1**	0.9756	0.9565	0.7857
$E_{trans}(m)$	0.0388	0.0583	0.0430	0.0954	0.0969	**0.0800**	0.0863	0.0863	0.1415
$E_{rot}(^\circ)$	8.6	5.1	6.1	6.3	2.7	**2.5**	12.2	13.7	14.9

2.5 Chapter Conclusion

In this chapter, three different methods for 3D object detection are implemented from a single RGB-D frame, namely the sample-score method, geometry method, and deep learning method.

The sample-score method can generate hundreds of 3D cuboid candidates by sampling object dimension and rotation, the candidates are then scored by image feature and physical constraints, and the best candidate is selected to represent the object. The geometry method converts the RGB-D image into 3D points, removes outliers, and cluster object points. On this basis, 3D cuboid can be generated to cover all corresponding points. The deep learning method learns a number of objects' properties from training data and predicts 3D cuboids of objects in different scenes. Regarding the framework, the individual loss function and joint loss functions are designed to encourage an accurate 3D estimation.

These three methods are evaluated on the indoor SUN RGB-D dataset as well as logistics IFL RGB-D dataset. In terms of speed, the sample-score method outperforms the other two methods, but the deep learning method achieves the highest accuracy. The geometry method provides a compromise between speed and accuracy. Besides, the relative object rotation and distance are two important factors that influence the object detection performance, which is useful for further application in a robotic platform.

3 Visual SLAM with Points, Planes, and Objects

This chapter presents a visual SLAM system at the level of objects, where feature points, structural planes, and semantic objects are jointly optimized in a unified graph. Section 3.1 introduces the background of visual localization and mapping. Section 3.2 reviews the related work of visual SLAM systems with different landmarks. Section 3.3 presents the framework of the proposed point-plane-objects SLAM system. Based on the state-of-the-art point-SLAM system, the system can detect planes and objects are detected in every key-frame, find the data association, and formulate an optimization problem with points, planes, and objects. In addition, an object-based loop detection module is added to make the system more robust. Section 3.4 evaluates the proposed method on indoor and logistics environments, the results show that the proposed SLAM system can improve camera localization and benefit scene understanding. Section 3.5 draws a conclusion for the chapter. This chapter's work is an extension to single image detection in Chapter 2.

3.1 Introduction

Visual SLAM refers to the problem of using image sequences to track the movement of a robot or a moving camera in an unknown environment and reconstruct the scene at the same time. Compared to other sensors like laser scanners, inertial units, or GPS, the camera has the advantages of a cheap price and low power consumption. Besides, it can provide robust and accurate environmental information

for scene understanding. Through years of development, visual SLAM has been applied in many fields, including virtual reality (Garon et al. 2016), autonomous driving (Lategahn et al. 2011), smart factory (Himstedt and Maehle 2017), etc.

Many visual SLAM systems take feature points as landmarks to track the movement of the camera, and the reconstructed maps are sparse and easy to update. However, these maps do not contain semantic or object information and are not convenient for high-level tasks such as robot manipulation. In man-made environments, many structures and objects carry semantics, and can potentially help reconstruct the map. For example, planes are good geometric elements to model the dominant structural layout of the environment (Li et al. 2021b). Objects can also provide semantic information to help understand the environment (Xia et al. 2020). Incorporating planes and objects will not only add landmarks to the map, but also provide additional constraints for SLAM optimization. Therefore, the proposed system integrates points, planes, and objects into the visual SLAM framework, explores the spatial relationship, designs the constraints, and jointly optimizes the whole system. Besides, in the localization and mapping process, there will be inevitable accumulated errors due to sensor noise (Wang et al. 2018). The loop detection module is designed to recognize if the place has been visited, which can effectively eliminate drift errors. Appearance-based loop detection methods rely on matching image intensities with local appearance-based features to recognize the same scene (Wang et al. 2020). These methods achieve impressive results under similar perceptual conditions, but have difficulties under large viewpoint changes. Semantic objects are robust to viewpoint variation, lighting conditions, and occlusion. So, an object-based loop detection is added alongside point-based loop detection to achieve global localization.

In this chapter, a point-plane-object SLAM system is proposed and four main challenges are addressed: landmark detection, data association, graph optimization, and loop detection. Building on the top of point-SLAM framework (Mur-Artal and Tardós 2017), structural planes and semantic objects can be detected from a single frame. Then, the spatial relationships among points, planes, and objects are exploited to establish the data association. These landmarks and associations can formulate a graph to optimize camera poses and other components. In addition, an

object-based loop detection module is designed to reduce estimation errors. The performance of the proposed method is evaluated on indoor and logistics environments. The results demonstrate that the proposed approach can estimate camera poses and simultaneously build a semantic map. In summary, the contributions can be described as follows:

- A visual SLAM system, where feature points, structural planes, and semantic objects are optimized within a unified bundle adjustment framework.
- A tracking and optimization method, which explores the spatial relationship among different landmarks and provides various constraints for optimization.
- A loop detection method, where an object-based graph matching strategy is used to detect loops under large viewpoints.
- Experiments on indoor and logistics environments, demonstrating the effectiveness of the point-plane-object SLAM system.

Table 3.1: Literature review of visual SLAM systems.

	Sensor	Point	Plane	Object	Loop Detection	Optimization
Davison et al. (2007)	RGB	image patch				point-point
Klein and Murray (2007)	RGB	FAST				point-point
Mur-Artal et al. (2015)	RGB	ORB			point	point-point
Hsiao et al. (2017)	RGB-D		plane		point	point-point, plane-plane
Zhang et al. (2019)	RGB-D		plane		point	point-point, plane-plane
Yang et al. (2016)	RGB	ORB	plane		point	point-point, plane-plane
Li et al. (2021b)	RGB-D	ORB	plane		point	point-point, plane-plane
Hosseinzadeh et al. (2017)	RGB-D	ORB	plane		point	point-point, plane-plane
Salas-Moreno et al. (2013)	RGB-D			3D model	point	point-point, object-object
Nicholson et al. (2018)	RGB			quadric	point	point-point, object-object
Liao et al. (2022)	RGB-D			quadric	point	point-point, object-object
Li et al. (2020a)	RGB			cuboid	point	point-point, object-object
Lin et al. (2021)	RGB			cuboid	point	point-point, object-object
Yang and Scherer (2019a)	RGB	ORB		cuboid	point	point-point, object-object, point-object
Hosseinzadeh et al. (2019)	RGB-D	ORB	plane	quadric	point	point-point, plane-plane, object-object, plane-object
Yang and Scherer (2019b)	RGB	ORB	plane	cuboid	point	point-point, plane-plane, object-object, plane-object

3.2 Literature Review

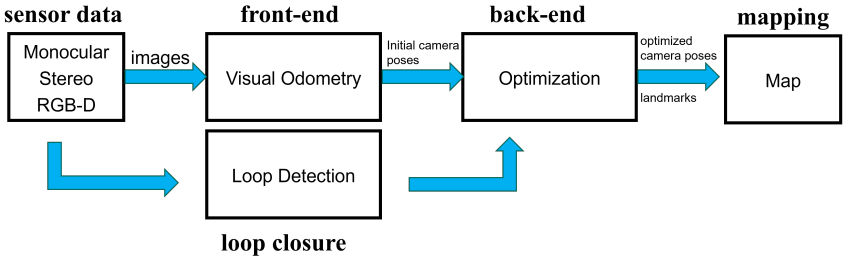


Figure 3.1: Overview of a typical visual SLAM system. A visual SLAM mainly includes four modules: front-end, back-end, mapping, and loop closure.

A visual SLAM system mainly includes four modules: front-end, back-end, mapping, and loop closure (Li et al. 2019a), as shown in Figure 3.1. The front-end module is also called visual odometry, which takes images as input, extracts features from the images, and estimates camera motion between adjacent images. The back-end accepts the camera poses measured by the front-end at different moments, as well as the information from loop detection, optimizes them, and obtains globally consistent trajectories and maps. The loop closure determines whether the camera has ever visited a previous position. If a loop is detected, it provides the information to the back-end for processing. Finally, the mapping module can reconstruct maps for different task requirements based on optimized trajectories.

Visual SLAM is an essential task for the autonomy of a robot, where the camera is the main source of external information. Depending on different cameras, visual SLAM can be divided into three categories: monocular, stereo, and RGB-D. Monocular SLAM uses a single RGB camera, stereo SLAM uses two, and RGB-D SLAM uses an RGB-D camera. Except for the sensors, visual SLAM can also be divided into feature-based and direct methods, where the direct methods use pixel intensities to establish correspondences, while the feature-based methods extract robust features to set up the association and create a map. Regarding different

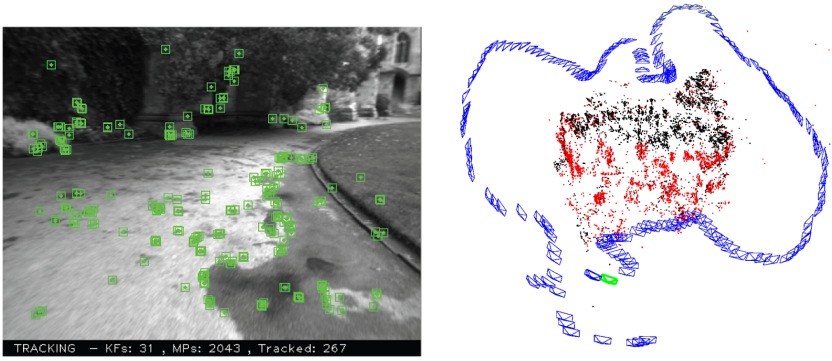
features, visual SLAM can also be divided into point-based SLAM, plane-based SLAM, object-based SLAM, and point-plane-object SLAM. Here the focus is on feature-based SLAM and some representative approaches are reviewed.

3.2.1 Feature Extraction and Point-based SLAM

A salient feature is a region of the image represented by its pixel position and appearance descriptor. One good-quality feature must be notable (easy to extract), precise (it may be measured with precision), and invariant to rotation, translation, scale, and illumination changes (Lemaire et al. 2007). To extract salient features, two phases are included: position detection and description extraction. The position detection process consists in generating a number of salient elements, such as corners. The description extraction consists in building a feature vector based on visual appearance near the pixel position.

With continuous developments in computer vision and robotics, some stable features are proposed, such as SIFT (Scale-Invariant Feature Transform) (Lowe 2004), SURF (Speeded Up Robust Features) (Bay et al. 2006), ORB (Oriented FAST and Rotated BRIEF) (Rublee et al. 2011), etc. Compared to other points, these features have the following properties: repeatability, distinctiveness, efficiency, and locality. SIFT is a robust feature that has rotation and scale invariance, and can solve the influence of noise and illumination changes. Lowe (2004) proposed to use a Gaussian convolution kernel to construct the multi-scale filtered image, then used an image pyramid to obtain refined feature points, and finally set the feature point direction according to the gradient direction histogram. However, due to its high dimension, SIFT requires a large number of calculations and cannot be realized in real time. On this basis, Bay et al. (2006) proposed SURF to improve efficiency. This descriptor has only 64 dimensions and can achieve 3 times the speed of SIFT. Rublee et al. (2011) proposed ORB feature, which consists of FAST (Features from Accelerated Segment Test) (Rosten and Drummond 2006) and BRIEF (Binary Robust Independent Elementary Features) (Calonder et al. 2010) descriptors. It accelerates the extraction process, reaching a speed of

30-50 times faster than SIFT and allowing real-time performance without GPU. Besides, it has comparative accuracy and provides good invariance to changes in viewpoint and illumination.



(a) ORB features **(b)** Point-based map

Figure 3.2: Example of point-based SLAM systems. They usually detect and track salient features (**a**) from images to estimate camera poses and simultaneously build a point-based map (**b**). The figures come from (Mur-Artal et al. 2015).

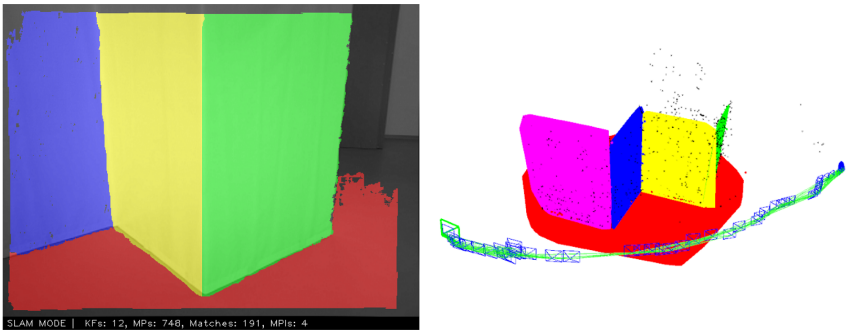
These feature points can be introduced into SLAM frameworks to realize localization. For example, Davison et al. (2007) proposed MonoSLAM, the first real-time monocular SLAM system that uses relatively large (11×11 pixels) image patches as long-term landmarks to make up the map. The detected features and estimated camera states are probabilistic and will be updated during camera motion and feature observation by the Extended Kalman Filter (EKF). Klein and Murray (2007) proposed to split tracking and mapping into two separate tasks and processed them in parallel threads on a dual-core computer: the tracking thread deals with the task of robustly tracking feature points, while the mapping produces a 3D map of point features from previously observed video frames. Although the above two systems are not robust and can only be applied in small scenarios, they are innovative and provide many pioneering ideas for modern SLAM systems. On this basis, Mur-Artal et al. (2015) proposed ORB SLAM, which is a complete

and reliable solution to visual SLAM: the front-end detects and tracks ORB features, the back-end uses non-linear optimization for camera pose estimation, the mapping thread takes ORB features to build the maps, and the same features can also be used for relocalization and loop closing, making the system more efficient, simple and reliable.

3.2.2 Plane Estimation and Plane-based SLAM

Planes are important geometric features in a structural environment, such as layout planes (wall, ground, etc.) and object surfaces (table surface, cabinet surface, etc.). To represent the planes, the point cloud model is widely used, which defines a plane as a group of points (Le and Košecka 2017). However, since one plane may contain hundreds of points, the optimization and data association operations require high computational costs regarding speed and memory. Another common approach to represent a plane is calculating the normal and distance to the origin. This representation provides clear geometric information, but it suffers from singularities and over-parametrized problems when involved in the optimization process (Taguchi et al. 2013). Besides, A homogeneous parametrization representation is proposed in (Kaess 2015), which is suitable for least-squares estimation with Gauss-Newton methods and incremental solvers. Therefore, the representation may be switched to another form according to different purposes. The point cloud model is used for reconstruction, the geometry model for description, and the homogeneous model for optimization.

Given a single frame, there are many approaches to estimate planes. An efficient way is to convert the depth image together with the camera's intrinsic parameters into a point cloud and segment it by applying plane estimation methods, such as RANSAC (Holz et al. 2011), region growing (Holz and Behnke 2013) or organized point cloud approaches (Trevor et al. 2013). On the other hand, it is also possible to estimate planes from a single RGB image. Hedau et al. (2009) treated the room layout as a bounding box and detected vanishing lines to model the planes. Yang et al. (2016) proposed to fit line segments along wall-ground boundaries and "pop



(a) Plane estimation **(b)** Plane-based map

Figure 3.3: Example of plane-based SLAM systems. They usually detect and track geometric planes (**a**) from images to estimate camera poses and simultaneously build a plane-based map (**b**). The figures come from (Hosseinzadeh et al. 2017).

up" a 3D model using geometric constraints. Both methods rely on the Manhattan world assumption (Coughlan and Yuille 1999), which states that all planes in the world are aligned with three dominant directions, typically corresponding to the X, Y, and Z axes. Furthermore, deep learning methods also offer some feasible solutions: PlaneNet (Liu et al. 2018) is a deep neural architecture for piece-wise planar estimation from a single RGB image and can directly infer a set of plane parameters. In addition, PlaneRCNN (Liu et al. 2019a), PlaneRecover (Yang and Zhou 2018), and PlaneSegNet (Xie et al. 2021) are some examples in this field.

When involved in a SLAM system, planes can act as landmarks to build a plane-based map and provide geometric constraints to improve camera localization. Hsiao et al. (2017) modelled planes as point clouds and proposed a key-frame based planar SLAM method to reconstruct dense indoor environments. Zhang et al. (2019) exploited the boundaries of planes and generated more virtual perpendicular planes, this method offers more constraints between planes to improve SLAM performance, but it also meets the problem of high computational cost. Combining with other geometric features, Li et al. (2021b) proposed to use points, lines, and planes to build an RGB-D SLAM for indoor environments, the result showed that this method can achieve a high localization and mapping accuracy under structured environments.

3.2.3 3D Object Detection and Object-based SLAM

3D object detection from a single frame remains a challenging problem because only 2D information is available. A complete review of single-frame object detection can be found in Section 2.2, here the focus is on object-based SLAM methods.

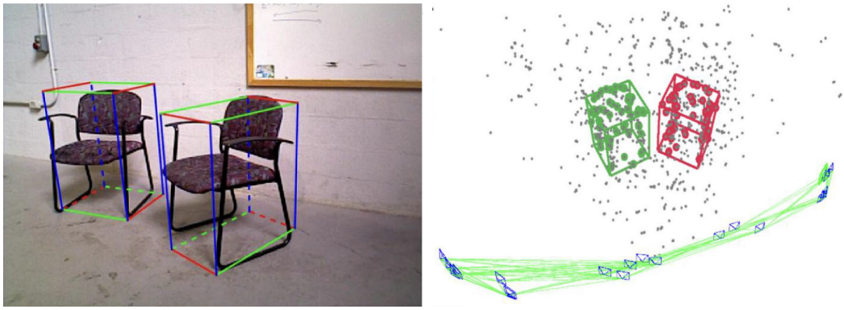
The introduction of objects in a SLAM system has a two-fold benefit. One is that objects can serve as geometric landmarks to improve localization accuracy. The other is that objects can benefit scene understanding because these semantic elements can enrich the reconstruction of the map.

SLAM++ (Salas-Moreno et al. 2013) is the first object-oriented SLAM, which used an RGB-D camera to reconstruct a cluttered scene with object models. It can recognize and track 3D objects in real-time, providing camera-object constraints to jointly refine the camera and object by efficient pose-graph optimization. QuadricSLAM (Nicholson et al. 2018) used dual quadrics as 3D landmark representations and incorporated quadrics into the SLAM framework. Their experiments showed that quadric landmarks not only provide valuable information for correcting odometry errors, but also significantly benefit the estimation of maps that contain objects as distinct elements. On this basis, quadrics can also be extended for symmetrical objects (Liao et al. 2022) and outdoor elements, such as (Meng and Zhou 2022), (Ok et al. 2019) and (Tian et al. 2021).

By representing objects as cuboids, Yang and Scherer (2019b) proposed a monocular object-based SLAM and demonstrated that object detection and SLAM benefit each other. On the one hand, SLAM improves the accuracy and robustness of 3D understanding, on the other hand, the object also benefits SLAM pose estimation and mapping. With a similar idea, Li et al. (2020a) presented to use objects for view-invariant loop detection and drift correction, resulting in an improvement in localization accuracy and robustness. Lin et al. (2021) modelled the environment as a topological graph with semantic objects and implemented object alignment between semantic maps. Experimental results demonstrated that the proposed

method can realize omnidirectional loop closure without viewpoint constraints, and is robust to environmental appearance changes.

Previous plane-based SLAM (Hosseinzadeh et al. 2017) and object-oriented SLAM (Salas-Moreno et al. 2013) systems focused either on planes or objects, but hardly considered both. The reason is that these high-level landmarks have different representations, making it challenging to explore the spatial relationship between them. To solve this problem, the proposed point-plane-object SLAM system represents all landmarks in the parametric form: feature points as pixels, planes as homogeneous vectors, and objects as cuboids. On this basis, the system exploits the data association problem among different landmarks and involves all landmarks in a unified bundle adjustment framework for localization and mapping.



(a) Object detection **(b)** Object-based map

Figure 3.4: Example of object-based SLAM systems. They usually detect and track semantic objects (**a**) from images to estimate camera poses and simultaneously build an object-based map (**b**). The figures come from (Yang and Scherer 2019b).

3.2.4 Data Association and Loop Detection

Data association is an important part of the SLAM system. Bowman et al. (2017) formulated a probabilistic data association for semantic SLAM, which estimated the data association distribution and maximized the expected measurement log-likelihood over the previously computed distribution. This method tightly coupled

inertial, geometric, and semantic observation into a single optimization framework.

Specifically, data association often happens in the tracking and optimization process. The point-based association is measured by appearance similarity. For example, SIFT and SURF calculate the Euclidean distance of the description vectors, and ORB points compute the hamming distance. Two points are matched if the feature distance is smaller than a threshold. In terms of geometric planes, the normal and relative distances are taken into consideration. Zhang et al. (2019) proposed that two associated infinite planes should have the same normal and are close to each other. Besides, for planes with boundaries, it is valuable to compare their intersection over the union. For object-object association, Li et al. (2020a) proposed to use a set of geometric and semantic information for the association, such as object translation, dimension, inliers, etc.

Then, data association among different features can also be found in some multi-landmark SLAM systems. For example, Hosseinzadeh et al. (2017) proposed a structure-aware SLAM system using quadrics and planes, where the quadrics and planes are estimated from RGB-D frames and provide a supporting affordance relationship between them. Yang and Scherer (2019b) presented a monocular object and plane SLAM in structured environments and exploited the spatial relationship between different landmarks.

Additionally, data association is also addressed in the loop detection module, because it needs to recognize if the landmarks have been previously viewed. The appearance-based methods extracted local or global visual features to find the association of images (Oliva and Torralba 2006), which convert global localization to an image retrieval problem. These methods showed good performance when the appearance difference between images is small. In some conditions, when the viewpoint is different, the appearance-based loop detections become less reliable. Instead, objects are robust to viewpoint variation, lighting conditions, and occlusion. This has inspired SLAM methods to use high-level object landmarks under large viewpoint differences. The graph-based methods formulated the object association as a graph registration problem, aiming to extract the correspondences

between nodes across the graphs. Gawel et al. (2018) introduced a random walks descriptor for every node that encodes the local connectivity of neighboring nodes. Based on the descriptor, the object match problem can be solved by computing the descriptor similarity between nodes. Liu et al. (2019c) combined random walk descriptor with graph matching and 3D alignment, the proposed object-level global localization algorithm is robust to illumination changes, view-point variation, etc. Lin et al. (2021) proposed an efficient graph matching method based on edit distance for robust place recognition, which demonstrated high accuracy and robustness against drastic scene and viewpoint variation.

3.3 Method

Section 3.3 is based on (Zhou et al. 2022). Parts of the following text are taken from that publication without changes.

In this chapter, a point-plane-object SLAM system is proposed on the top of ORB-SLAM2 (Mur-Artal and Tardós 2017). As shown in Figure 3.5, in every key-frame, points, planes, and objects are detected (see Section 3.3.1). Next, these landmarks should be tracked among multiple key-frames with a data association strategy (see Section 3.3.2). Then, the associated landmarks will form a graph to jointly optimize all components (see Section 3.3.3). An object-based loop detection module is added alongside point-based loop detection to achieve global localization (see Section 3.3.3). Finally, camera poses can be updated and a point-plane-plane map can be reconstructed.

3.3.1 Features Detection

Point Extraction. The system rely on ORB-SLAM 2 (Mur-Artal and Tardós 2017) implementation to extract ORB feature points, which are represented as 3D points with descriptors. The positions of feature points are represented as $\mathbf{P} = (x, y, z, 1)^\top$.

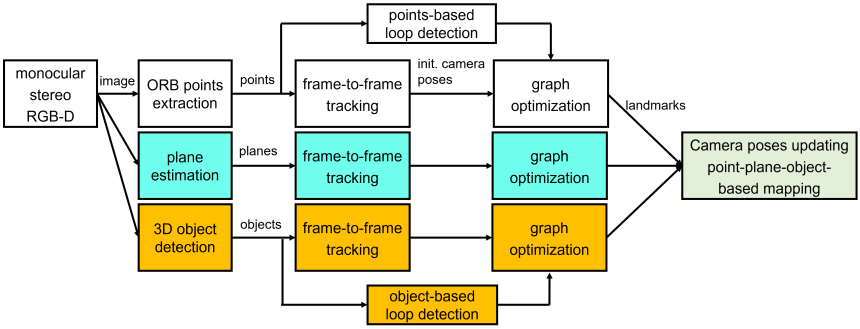


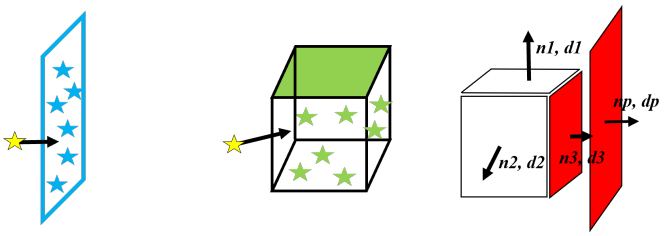
Figure 3.5: Overview of the proposed point-plane-object SLAM system. It builds the point-plane-object SLAM on top of ORB-SLAM2. In each key-frame, it detects planes and objects, tracks them across frames, adds them to a unified graph, and jointly optimizes all the components. An object-based loop detection module is added to make the whole system more robust. As a result, camera poses can be updated and a point-plane-plane map can be reconstructed.

Plane Estimation. Given a key-frame, it can be converted into a 3D point cloud and a multiple plane estimation method (Trevor et al. 2013) is applied to estimate all planes in the scene. Under the Manhattan world assumption (Coughlan and Yuille 1999), under the assumption that planes are parallel or perpendicular to each other and filter planes with regard to three main directions. The planes are detected as $\boldsymbol{\pi} = (n_x, n_y, n_z, d)^\top$, and will convert into homogeneous vectors $q(\boldsymbol{\pi}) = (\phi, \psi, d)^\top$ for optimization.

Object Detection. As discussed in Section 2.3, the sample-score method is adopted to detect 3D objects from RGB images. In this chapter, objects are represented with the parametric form for a unified optimization process, where objects are defined as $\mathbf{O} = (\mathbf{T}_o, \mathbf{D})$, where $\mathbf{T}_o$ and $\mathbf{D}$ are the pose and dimension.

3.3.2 Data Association

Data association means to find the relationship among different features, it is an important part of the SLAM system that happens in single-frame detection, across frame tracking, and frame-map matching.



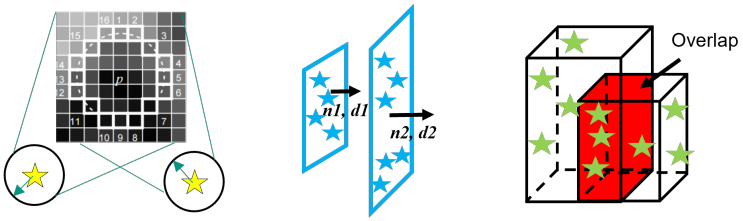
(a) Point-plane association. **(b)** Point-object association. **(c)** Plane-object association.

Figure 3.6: Proposed data association strategy of the SLAM system in key-frame. (**a**) Points are associated with planes by distance threshold. (**b**) Points are associated with objects by distance threshold. (**c**) Planes are associated with objects by normal and distance thresholds.

For every key-frame, feature points, planes, and objects are detected and associated. As illustrated in Figure 3.6, the point-plane association is computed by the orthogonal distance between them. Points are selected as in-plane points $\mathbf{P_{plane}}$ when they are observed in at least 3 key-frames and their orthogonal distance satisfies a threshold $d_{\pi p}$. For point-object association, the method checks if the pixel position of points lies inside the object's 3D bounding box. If a feature point is observed in more than 3 key-frames and lies inside an object bounding box, it will become in-object points $\mathbf{P_{object}}$. Finally, the method associates the plane and object by checking if their orientation and orthogonal distance meet the angle threshold $\theta_{\pi o}$ and distance threshold $d_{\pi o}$.

The system also consider landmark tracking across frames. As shown in Figure 3.7, the feature points are matched by the descriptor (Mur-Artal and Tardós 2017). For plane-plane association, three conditions are required: the orientation threshold $\theta_{\pi\pi}$, the orthogonal distance threshold $d_{\pi\pi}$, and the minimum number of in-plane points $N_{\pi\pi}$. For object-object association, two objects that satisfy a 3D IoU threshold IoU_{3D} and share a minimum number of in-object points N_{oo} are regarded as associated.

To reduce visual odometry drift, a point-plane-object map is initialized and maintained. For every key-frame, mapped landmarks are transferred to the current frame and matched to the frame-detected landmarks. The thresholds are the same



(a) Point-point association. **(b)** Plane-plane association. **(c)** Object-object association.

Figure 3.7: Proposed data association strategy of the SLAM system across frames. (**a**) Points are associated by description matching. (**b**) Planes are associated by the normal, distance, and in-plane points. (**c**) Objects are associated by overlapping and in-objects points.

as across frame tracking. If the detected landmarks satisfy these conditions, they will be associated and optimized by unified bundle adjustment. Otherwise, they will be initialized as new landmarks and added to the map.

3.3.3 Graph Optimization

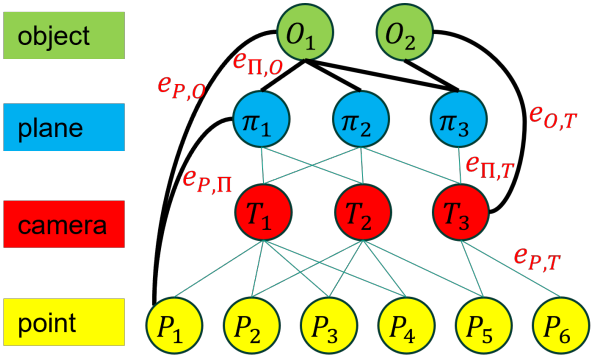


Figure 3.8: Proposed optimization graph of the SLAM system, where the red nodes with T denote the camera key-frames, while the green nodes with O, blue nodes with π and yellow nodes with P denote the objects, planes, feature points landmarks respectively. The edges represent the constraints between them.

After association, a unified graph can be constructed to represent the SLAM problem. As shown in Figure 3.8, every node in the graph corresponds to a pose of the robot or a location of landmarks, while every edge between two nodes corresponds to a constraint between them. With the help of the graph, the SLAM problem turns to find a node configuration that minimizes the error introduced by the constraints. Mathematically, the graph optimization process can be formulated to find the minimal value of the following nonlinear least squares problem:

$$C^*, O^*, \Pi^*, P^* = \underset{\{C,O,\Pi,P\}}{\arg\min} \sum_{i\in C, j\in O, k\in\Pi, m\in P} \mathbf{e}^T \mathbf{\Sigma} \mathbf{e}, \tag{3.1}$$

where $C = \{T_{ci}\}$, $O = \{o_j\}$, $\Pi = \{\pi_k\}$ and $P = \{p_m\}$ represent camera poses, cuboids, planes, feature points respectively. $\mathbf{\Sigma}$ is a covariance matrix of different constraints. The problem can be solved by the Gauss–Newton or Levenberg–Marquardt algorithms available in many libraries, such as g2o (Kümmerle et al. 2011). More details about the optimization problem can be found in Appendix B. The constraint errors between different landmarks are represented by $\mathbf{e}$ and will be explained as follows:

1) Camera-Point Constraint: The proposed method follows ORB-SLAM2 (Mur-Artal and Tardós 2017) to minimize the geometric re-projection error as:

$$e(\mathbf{P}, \mathbf{T}_c) = \mathbf{u}_c - \rho(\mathbf{T}_{cw}^{-1}, \mathbf{P}_w), \tag{3.2}$$

where $\mathbf{u}_c$ is the image pixel location of the 3D point $\mathbf{P_w}$ in the global world frame, $\rho(.)$ is a function that projects a world 3D point into the image plane.

2) Camera-Plane Constraint: Since the homogeneous representation of the plane $\boldsymbol{\pi} = (n_x, n_y, n_z, d)^\top$ will cause an over-parameterization in the optimization problem (Zhang et al. 2019), it is converted into a minimal representation $q(\boldsymbol{\pi}) = (\phi, \psi, d)^\top$ as:

$$q(\boldsymbol{\pi}) = \left(\phi = \arctan\frac{n_y}{n_x}, \psi = \arcsin n_z, d\right)^\top, \tag{3.3}$$

where ϕ and ψ are the azimuth and elevation angle of the plane normal respectively and restricted to $(-\pi, \pi]$ to avoid singularities. Then, the camera-plane constraint can be formulated as:

$$e(\boldsymbol{\pi}, \mathbf{T}_c) = q\left(\boldsymbol{\pi}_c\right) - q\left(\mathbf{T}_{cw}^{-\top} \boldsymbol{\pi}_w\right), \tag{3.4}$$

where $\boldsymbol{\pi}_c$ is the observed plane in the current frame, $\mathbf{T}_{cw}^{-\top} \boldsymbol{\pi}_w$ means to transform 3D planes $\boldsymbol{\pi}_w$ from world coordinates to current frame.

3) Camera-Object Constraint: Since objects can be regarded as 8 vertices, the camera-object constraint can be converted to the camera-point constraint. In other words, the cuboid landmark $\mathbf{O}_w$ is projected onto the image plane to get 8 vertices, then the camera-object constraint can be calculated by the sum of the geometric projection of 8 corners $\mathbf{z}_m$ as:

$$e(\mathbf{O}, \mathbf{T}_c) = \sum_{m=1}^{8} \mathbf{z}_m - \rho(\mathbf{T}_{cw}^{-1}, \mathbf{O}_w). \tag{3.5}$$

4) Point-Plane Constraint: If a feature point $\boldsymbol{P}$ lies on a specific plane $\boldsymbol{\pi}$, the point-plane constraint is defined as the orthogonal distance from that point $\boldsymbol{P}$ to its associated plane $\boldsymbol{\pi}$:

$$e(\mathbf{P}, \boldsymbol{\Pi}) = |\mathbf{n}_\pi \mathbf{P}| - d_\pi. \tag{3.6}$$

5) Point-Object Constraint: If a point $\boldsymbol{P}$ belongs to an object, it should lie inside the 3D bounding box. The method transforms the point into the associated cuboid frame and compares its location with the cuboid's dimensions, defined in (Yang and Scherer 2019a):

$$e(\mathbf{P}, \mathbf{O}) = \max\left(\left|\mathbf{T}_o^{-1} \mathbf{P}\right| - \mathbf{D}, \mathbf{0}\right), \tag{3.7}$$

the $\max$ operator is used because the points are only encouraged to lie inside the cuboid instead of the surface.

6) Plane-Object Constraint: Two different types of planes are associated with objects. One is the structural plane, such as the ground or the wall. They are parallel to the world structure and provide supporting constraints for objects. The other plane type is the object surface plane, which belongs to the object and should be close to the cuboid proposal. These different planes can be assigned with individual weights. In this chapter, they are the same.

Since an object is represented by a cuboid, the six surface planes $\boldsymbol{\pi_{oi}}, i \in \{1, 6\}$ can be defined by the center position, orientation, and dimension of the object. If a plane $\boldsymbol{\pi}$ is associated with the object, as shown in Figure 3.6, it should have a similar angle and minimum distance to one of the cuboid surface planes. Then, the plane-object constraint can be replaced by the plane-plane constraint and calculated as:

$$e(\mathbf{\Pi}, \mathbf{O}) = \min_{i \in [1,6]} (q(\boldsymbol{\pi}) - q(\boldsymbol{\pi}_{oi})). \tag{3.8}$$

3.3.4 Loop Detection

Section 3.3.4 is based on (Zhou et al. 2022). Parts of the following text are taken from that publication without changes.

After tracking and optimization, the camera poses can be updated, and a point-plane-object map can be reconstructed. However, due to sensor noise, there might be some accumulated errors, they should be reduced by loop detection modules. A typical example is shown in Figure 3.9, in this case, point-based loop detection doesn't work under large viewpoint changes, but it can be solved by object-based loop closure to achieve global localization. The key idea is to use topological information.

The whole process is illustrated in Figure 3.10 and described as follows: To detect object-based loops, all objects are manually divided into two groups according to their index in the map, as the most recently mapped landmarks are matched to earlier landmarks. Let L_a and L_b be two landmark groups; an attempt is made to

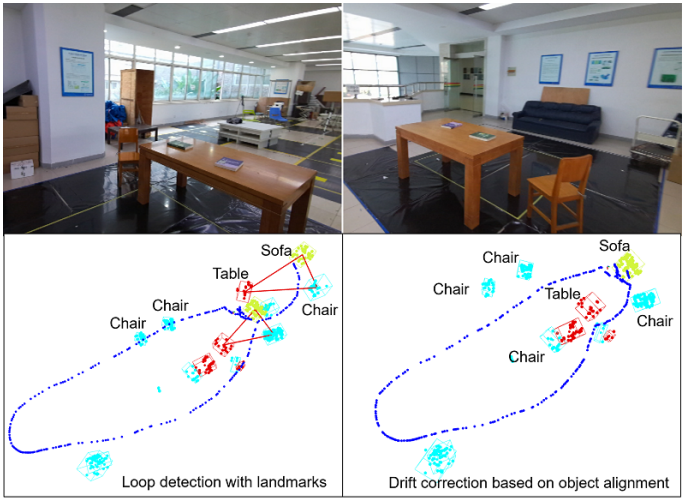


Figure 3.9: Proposed object-based loop detection of the SLAM system, An example of the SLAM system with object-based loop closure. Top: The Same objects are observed from two different viewpoints. Left: An object-based loop is detected by graph matching. Right: The drift error is corrected by object alignment.

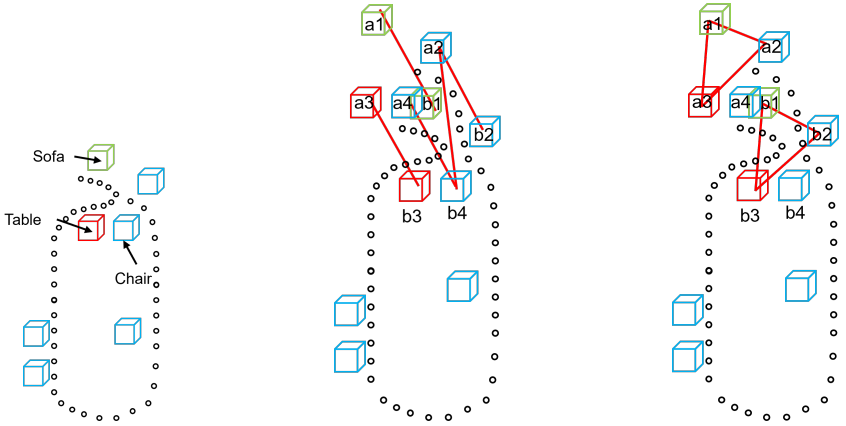
identify a subgroup of L_a that matches the nodes of L_b. To avoid false matches, each candidate pair is checked to meet the following verifications.

General verification: Landmarks that have the same label and great closeness are the only ones considered. To measure the closeness of nodes a and b, the index separation is defined as:

$$\eta(a,b) = |ID_a - ID_b| \ (a \in L_a, b \in L_b), \tag{3.9}$$

A closeness threshold is define $c\_thre = 3$, if $\eta(a,b) > c\_thre$, a and b are considered as loop candidates, otherwise, they should be merged by data association.

Topological verification: By representing the object-oriented map as a topological graph, each object can be described with its surrounding information, such as neighbor vector descriptor (Stumm et al. 2016) and semantic histogram descriptor (Guo et al. 2021). Inspired by these, a directional histogram descriptor is proposed



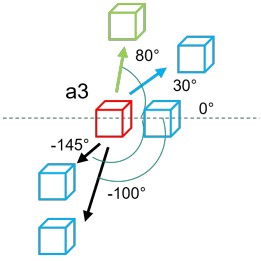
(a) The ground truth of object-based map.

(b) The object-based map with topological verification.

(c) The object-based map with geometry verification.

Figure 3.10: Example of object-based loop detection (**a**) The ground truth. (**b**) Objects are connected if they satisfy the topological verification. (**c**) Objects are connected if they satisfy the geometry verification.

to find object association. To be specific, for each node, it defines the direction angle range by splitting 2π into $div(div = 8)$ parts and counts all neighboring objects regarding the relative angle and the label. An illustration of the descriptor is shown in Figure 3.11.



(a) The topological graph.

Yaw Angle Range	Chair	Sofa	Table
[-180°, -135°]	1	0	0
[-135°, -90°]	1	0	0
...	...	...	...
[0°, 45°]	2	0	0
[45°, 90°]	0	1	0
[90°, 135°]	0	0	0
[135°, 180°]	0	0	0

(b) The directional histogram descriptor.

Figure 3.11: Proposed directional histogram descriptor for loop detection. (**a**) Take a3 as an example to draw the topological graph. (**b**) Take a3 as an example to generate the directional histogram descriptor.

To verify two objects have similar graph structures, their directional histogram descriptors H are compared, and the similarity score $T_{score}(a, b)$ is computed. The score is calculated by the normalized dot-product between two descriptors as follows:

$$T_{score}(a,b) = \frac{\sum_{d=1}^{n_d} H_a \times H_b}{\sqrt{\sum_{d=1}^{n_d} (H_a)^2} \times \sqrt{\sum_{d=1}^{n_d} (H_b)^2}}, \tag{3.10}$$

where n_d denotes the descriptor size and is equal to angle range parts div multiple semantic label size ($div \times n = 8 \times 6 = 48$). coarse matches can be obtained if the similarity score is greater than the descriptor similarity threshold. ($des\_thre = 0.8$).

Geometry verification: The coarse matching candidates still contain many incorrect matches. For example, in Figure 3.10(**b**), $a3$ is connected to $b4$. In graph-based localization, the transformation between two graphs is rigid, and it has been shown that given at least 3 matched objects can recover the relative transform (Li et al. 2019b). Sub-graphs are established as (a_1, a_2, a_3) and (b_1, b_2, b_3) from coarse matches, and it is checked if the connected edges have scale consistency. The scale ratio (G_{score}) is measured as:

$$G_{score}(A,B) = \frac{|a_1, a_2|}{|b_1, b_2|} \approx \frac{|a_2, a_3|}{|b_2, b_3|} \approx \cdots \approx \frac{|a_{n-1}, a_n|}{|b_{n-1}, b_n|} \tag{3.11}$$

where $|a_1, a_2|$ represents the distance between a_1 and a_2. False object correspondences often lead to an unreasonable scale ratio and have a large scale error. So, only if the scale error is smaller than a scale threshold ($s\_thre = 0.01$), these two sub-graphs are marked as inlier matches.

Duplication verification: All possible sub-graph combinations of coarse matches are traversed, which may cause duplications in different inlier matches. So, the last process is to check all inlier matches, add them to final matches, and remove

the duplication if they exist. Having at least 3 final matches can be recognized as a loop, and the corresponding landmark information is extracted for loop correction.

Loop correction: Once a candidate is accepted as a valid loop, it can be used for drift correction by the object alignment. Here, a similarity transformation consisting (rotation R, translation t, and scaling s) is computed where objects from group L_a can be mapped to corresponding objects from group L_b as: $O_a = sR * O_b + t$. In fact, only using the object centroid may cause large errors, instead, the method takes the associated map points and applies a scaled ICP (Zinßer et al. 2005) to calculate the similarity transform. Given two looped sub-graphs $(a_1, a_2, \cdots, a_n)$ and $(b_1, b_2, \cdots, b_n)$, the scaling is first computed using all matched object centroid:

$$Scale(A, B) = \frac{1}{n} \sum \frac{|a_1, a_2|}{|b_1, b_2|} + \frac{|a_2, a_3|}{|b_2, b_3|} + \cdots + \frac{|a_{n-1}, a_n|}{|b_{n-1}, b_n|}. \tag{3.12}$$

Then, the method scales all map points from sub-group $(b_1, b_2, \cdots, b_n)$, obtains inlier map point correspondences, and computes the rigid transform. The rotation and translation are calculated by minimizing the sum of the squared error:

$$(R^*, t^*) = \underset{R,t}{\operatorname{argmin}} \sum_{k=1}^{N_p} \left\| P_{a,k} - sR * P_{b,k} - t \right\|^2, \tag{3.13}$$

where $P_{a,k}$ and $P_{b,k}$ are the correspondence map points of two sub-graphs after RANSAC rejection, and N_p is the matched points number. After obtaining the similarity transform, the proposed method performs the transform to the recently mapped landmarks, merges the duplicated objects, and corrects the camera poses that can observe these objects. Finally, the whole semantic map is updated with non-linear least square optimization (see Section 3.3.3). More details and related results can be found in (Zhou et al. 2022).

3.4 Experiments

Section 3.4 is based on (Zhou et al. 2022). Parts of the following text are taken from that publication without changes.

The performance of point-plane-object SLAM is evaluated on the ICL-NUIM dataset (Handa et al. 2014) because it provides rich plane and object information. As shown in Figure 3.12, two different scenarios, namely home and office scenarios with 8 sequences are included in the dataset, where the home scenario contains $sofas$, $chairs$, and $vases$, while the office scenario contains $monitors$, $tables$, $cabinets$, etc.

Figure 3.12: ICL-NUIM dataset. This dataset provides several RGB-D sequences of different indoor scenes with object instances. The figure comes from the official website: https://www.doc.ic.ac.uk/ ahanda/VaFRIC/iclnuim.html.

In the point-plane-object SLAM, object detection and plane estimation only happen at key-frame and do not cost much computation resources. To ensure a robust SLAM system, a strict outlier rejection is added to data association, where the distance threshold is defined as $d = d_{\pi p} = d_{\pi o} = d_{\pi\pi} < 0.1m$, angle threshold $\theta = \theta_{\pi o} = \theta_{\pi\pi} < 8°$, IoU threshold $IoU_{3D} > 0.5$, the minimum number of

in-plane points and in-object points $N = N_{\pi\pi} = N_{oo} > 15$. These values guarantee a stable performance for the ICL-NUIM dataset. When applied to other environments, these parameters may be different.

The root mean squared error (RMSE) of absolute pose error (APE) [1] are adopted as metrics to evaluate the localization performance. For comparison, baseline experiments are performed against ORB-SLAM2 (Mur-Artal and Tardós 2017) and benchmark the proposed point-plane-object SLAM system against state-of-the-art ORB-SLAM3 (Campos et al. 2021).

Except for open-source datasets, experiments are also designed on a real robotic platform to estimate the SLAM performance. The mobile robot is manually steered along a trajectory in a logistics environment to realize self-localization and semantic mapping.

3.4.1 Experiments on Indoor Environments

1) Qualitative Results: An example of the point-plane-object SLAM in living room sequences kt-2 is shown in Figure 3.13. On the left side, the first column shows the single-frame object detection result on RGB images, and the camera trajectory and the reconstructed map are shown on the right. Taking points, planes, and objects into a unified local mapping framework, the system can build a point-plane-object map. It can be seen that 5 objects and 9 planes are visualized, where the associated points, planes, and objects share the same color, while other unassociated planes such as ceilings are not displayed. Note that in the SLAM optimization, planes are represented as infinite planes, but for visualization purposes, the point cloud model is used.

For office sequences, similar experiments are done as for the living room sequences. Figure 3.15 shows the detection and mapping result on sequence kt-0 as an example, where 6 objects and 5 associated planes are visualized. Compared

[1] https://github.com/MichaelGrupp/evo

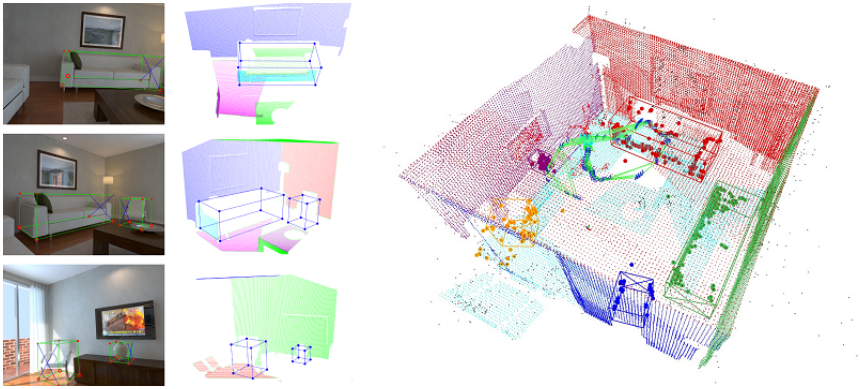


Figure 3.13: SLAM results on living room kt-2 sequence of ICL NUIM dataset, where the left part shows the object detection results in image and 3D space, while the right part shows the camera trajectory (green line) and the reconstructed point-plane-object map. The associated points, planes, and objects share the same color, and other unassociated planes such as ceilings are not displayed. The point cloud model is utilized as the plane representation only for visualization.

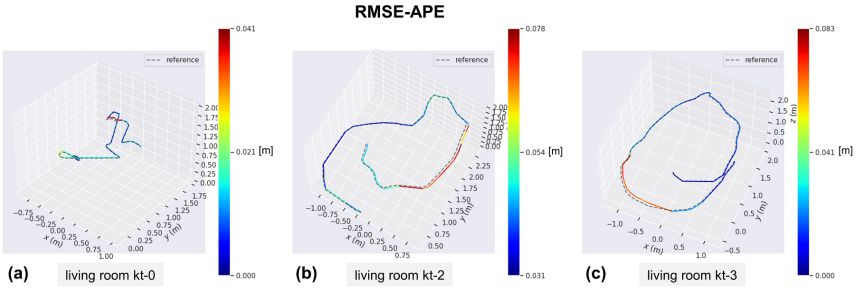


Figure 3.14: Evaluation on living room sequences of ICL NUIM dataset, these figures show the comparison of the estimated trajectories and corresponding ground truth.

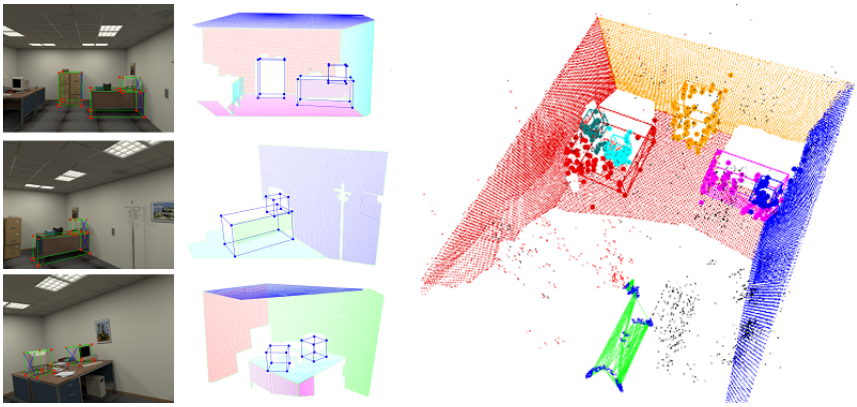


Figure 3.15: SLAM results on office kt-0 sequence of ICL NUIM dataset, where the left part shows the object detection results on image and 3D space, while the right part shows the camera trajectory (green line) and the reconstructed point-plane-object map.

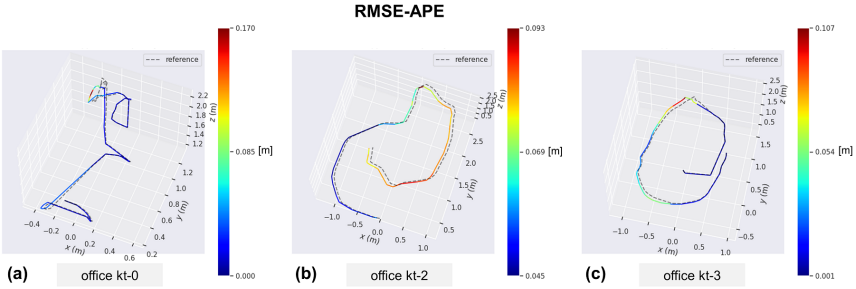


Figure 3.16: Evaluation on various office sequences of ICL NUIM dataset, these figures show the comparison of the estimated trajectories and corresponding ground truth.

to the living room dataset, office sequences include objects that are not located on the ground. For example, the monitors are on the top of the desk, and the object detection method also works. It should be pointed out that in the third row of Figure 3.15, there are sometimes detection errors from the single image due to the clustered background or occlusion. These imperfect object constraints are optimized in the SLAM framework by sacrificing localization accuracies. Since there are only less than 10 objects but hundreds of feature points, even if objects do not improve the results, they won't seriously damage the system.

2) Quantitative Results: Experiments are conducted on other sequences of the dataset, the localization performances with RMSE-APE between estimated camera pose and ground truth are shown in Figure 3.14 and Figure 3.16, it can be seen that the proposed method reaches a localization accuracy between $0.01m$ and $0.05m$, the mapping results with landmark information, including the key-frame number, object number, and plane number, are reported in Table 3.2 and visualized in Figure 3.17 and Figure 3.19. Again, the propsoed SLAM system can achieve good localization (RMSE-APE < 0.170 m) and mapping results (sparse semantic maps).

Table 3.2: Landmark information of the point-plane-object map on ICL NUIM Dataset.

Sequence	Frame Number	Key-Frame Number	Object Number	Plane Number
livingroom kt-0	1508	43	1	10
livingroom kt-2	880	57	6	16
livingroom kt-3	1240	120	0	14
office kt-0	1508	57	6	7
office kt-2	880	58	7	10
office kt-3	1240	62	1	8

For comparison, ORB-SLAM3 (Campos et al. 2021) is implemented in these sequences. To evaluate the benefit of introducing planes and objects, two variations

of the system are also studied, namely point-plane SLAM and the point-object SLAM system. For camera trajectory comparison, the results are plot in Figure 3.18 and Figure 3.20. Since the ORB feature-based initialization (Mur-Artal and Tardós 2017) relies on RANSAC, each system runs 5 times in each sequence, and the mean value of RMSE-APE is reported in Table 3.3. It can be observed that the added object and plane landmark constraints in the proposed method improve the camera pose estimation. The improvement comes from two aspects: 1) the plane detection with dominant direction contributes to better pose estimation, because there are significant improvements between point SLAM and point-plane SLAM. 2) the strict outlier rejection method ensures a robust data association, and the constraints between different landmarks encourage a reliable and precise result. However, in sequence kt-1, both ORB-SLAM3 and the proposed system do not work and are therefore not shown in the table. In sequence kt-3, when no objects are detected, the point-plane-object SLAM is reduced to point-plane SLAM, the results are different because of RANSAC initialization.

Table 3.3: Evaluation of camera absolute pose error root mean squared error (RMSE-APE) on ICL NUIM dataset (cm).

Sequence	Point-based SLAM (ORB-SLAM2)	Point-based SLAM (ORB-SLAM3)	Point-Plane SLAM	Point-Object SLAM	Point-Plane-Object SLAM
livingroom kt-0	**0.39712**	0.75052	0.49278	0.40216	0.99162
livingroom kt-2	2.57732	2.17204	2.3053	2.0543	**1.77044**
livingroom kt-3	2.64344	3.16458	2.0557	2.55202	**1.75516**
office kt-0	12.83628	6.31114	**5.85264**	6.32358	6.83028
office kt-2	2.61264	1.86694	**1.42606**	5.3889	1.7747
office kt-3	3.92022	4.18088	**2.90242**	3.11014	3.66788

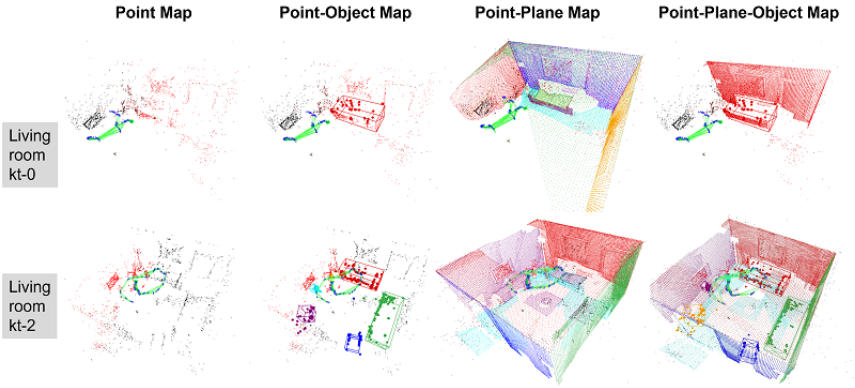


Figure 3.17: Comparison of the mapping results using different SLAM systems on various living room sequences of ICL NUIM dataset. Note that in point-plane-object map, only associated planes are visualized.

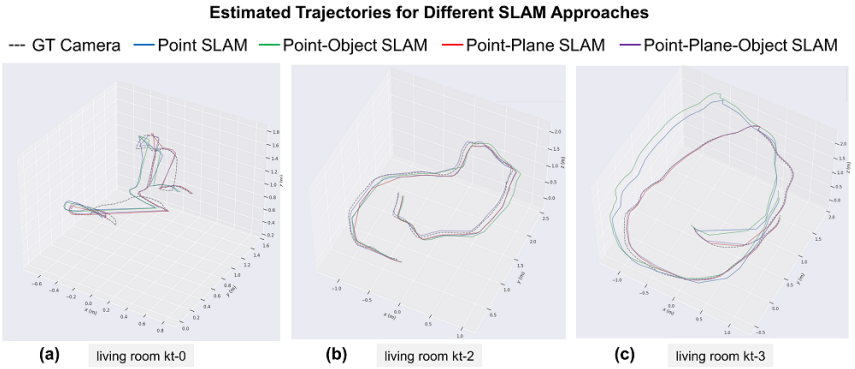


Figure 3.18: Comparison of the camera trajectories using different SLAM systems on various living room sequences of ICL NUIM dataset.

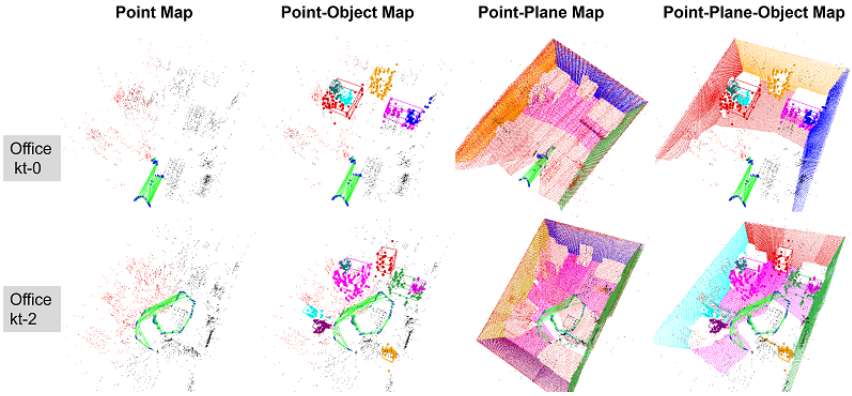


Figure 3.19: Comparison of the mapping results using different SLAM systems on various office sequences of ICL NUIM dataset. Note that in point-plane-object map, only associated planes are visualized.

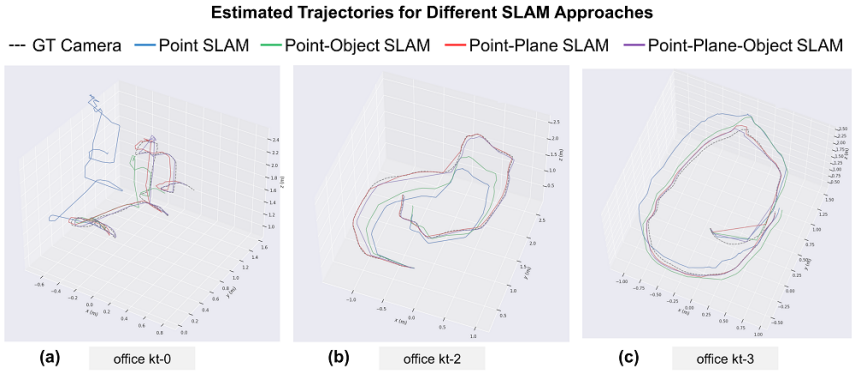


Figure 3.20: Comparison of the camera trajectories using different SLAM systems on various office sequences of ICL NUIM dataset.

3) Runtime Analysis: Mapping and localization are usually conducted locally on a mobile robot, therefore runtime is a critical factor for real-world systems with limited computing resources, and the SLAM runtime experiments are conducted on the ICL NUIM living room kt-2 sequence. All methods are implemented in C++ and evaluated on a laptop computer (i7-8565U 1.80 GHz CPU, 16 GB RAM, no GPU, Ubuntu 18.04). As shown in Table 3.4, the object detection on RGB image requires 725 ms in the test machine. The tracking thread, including feature detection and data associations in every key-frame, needs an average of 207 ms. Then, it takes 197 ms for local bundle adjustment with points, planes, and objects. Compared to point-only bundle adjustment, the proposed system has higher optimizing costs because point-plane and point-object association have to be applied to many points, increasing the optimization time. A demo video [2] is provided to illustrate the whole process of SLAM.

Table 3.4: Evaluation of runtime performance on ICL NUIM dataset.

Tasks	Runtime (mSec)
Object Detection	725.05
Tracking Thread*	207.376
Point Only BA	63.240
Point Plane Object BA	197.48

* The tracking thread includes feature extraction, plane estimation, and data association.

3.4.2 Experiments on Logistics Environments

Different from indoor environments, logistics scenes are designed to facilitate the efficient movement of goods and materials, they always have a large space and different features. AgiProbot project (Klein et al. 2021) is a typical intra-logistics

[2] https://github.com/benchun123/point-plane-object-SLAM

system in an agile remanufacturing production system, the mobile robot is tasked to deliver items among different workstations, thus needs a localization system and a semantic map for navigation. As discussed in this chapter, the proposed point-plane-object SLAM provides a suitable solution for this challenge.

To achieve this goal, a Microsoft Azure Kinect camera and a 2D SICK laser scanner are mounted, as shown in Figure 3.21. The camera is installed at a height of $1.5m$, facing downwards at $45°$ to capture RGB and depth images. In the experiment, the laser scanner is used for ground truth generation and safety. Calibration between the camera and base link is solved by (Zhang and Pless 2004). All codes are implemented with C++ on an industry computer (i5-8565U 1.80 GHz CPU, 16 GB RAM, no GPU, Ubuntu 16.04).

(a) The robotic platform. **(b)** The AgiProbot environment.

Figure 3.21: The robotic platform and environment in AgiProbot project.

The vehicle was manually steered along a trajectory to map the whole environment $(6 * 12m)$. Since the camera poses are not available, the laser scan is used to generate the ground truth. To do so, ICP registration (Censi 2008) is adopted to estimate the movements of the laser scanner and convert them to the ground truth of camera poses. For object detection, $conveyors$ on the top of the workstation are selected as object landmarks because they can act as the goal for navigation. To detect the object, 100 images with $conveyors$ are captured, label them manually

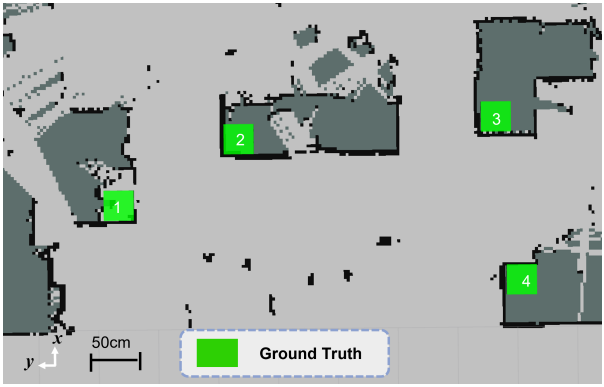


Figure 3.22: The object ground truth in the logistics environment. It uses laser scan data to build a 2D occupancy map, based on the map, camera trajectory can be generated as the ground truth. conveyor positions are also labelled on the map.

as training data, and feed them to YOLOv2 (Redmon and Farhadi 2017) network to obtain pre-trained weight for the object detection task. The object positions are annotated in the map and shown in Figure 3.22.

Similar to the experiments on indoor datasets, the root mean squared error (RMSE) of absolute pose error (APE) is adopted as a metric to evaluate the SLAM performance. For comparison, the proposed point-plane-object SLAM system is compared against point-based SLAM, point-plane SLAM, and point-object SLAM.

Figure 3.23 shows the detection and mapping results of the whole environment, where 4 objects and 11 planes are involved. It can be observed that the proposed point-plane-object SLAM method can map the logistics environment with geometric and semantic landmarks, which can benefit scene understanding.

The localization performances with RMSE-APE are shown in Figure 3.24, which reaches an accuracy of $0.2m$ when compared with laser-based ground truth, showing a worse performance than indoor datasets. It can be explained by the large environment and camera movement. On one hand, the camera has a detection range of $0 \sim 3m$, while the test environment is much larger than this range, points over this will introduce errors to the localization system. On the other

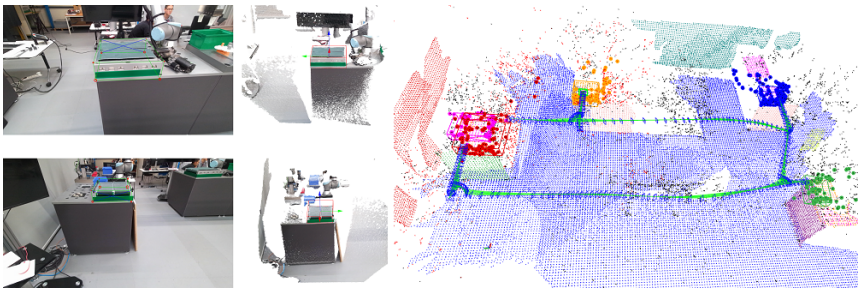


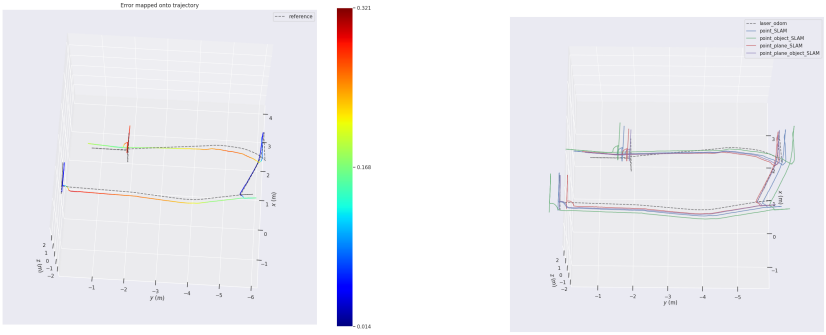
Figure 3.23: SLAM result on the logistics environment. where the left part shows the object detection results in image and 3D space, while the right part shows the camera trajectory and the reconstructed point-plane-object map. It utilize the point cloud model as the plane representation only for visualization.

Table 3.5: Evaluation of camera absolute pose error root mean squared error (RMSE-APE) on a logistics environment (m).

Methods	RMSE-APE (m)
Point-based SLAM	0.2559
Point-plane SLAM	0.2174
Point-object SLAM	0.2635
Point-plane-object SLAM	**0.2055**

hand, the scenario is monotonous, fewer feature points are detected than indoor environments. Furthermore, the movements of the camera (estimated) and laser scan (ground truth) are different, especially when the robot rotating, which brings errors when estimating the localization accuracy.

For comparison, a quantitative result on camera localization is compared in Table 3.5, and the corresponding mapping results are shown in Figure 3.25. When compared to other visual SLAM methods, the proposed point-plane-object SLAM is better than others, proving that the introduction of more landmarks can improve localization accuracy.



(a) Evaluation of SLAM with RMSE-APE.

(b) Comparison of camera trajectories.

Figure 3.24: Evaluation and comparison of the point-plane-object SLAM system on the logistics environment. (**a**) This figure shows the comparison of the estimated trajectories and corresponding ground truth. (**b**) This figure shows the comparison of the estimated trajectories using different SLAM systems.

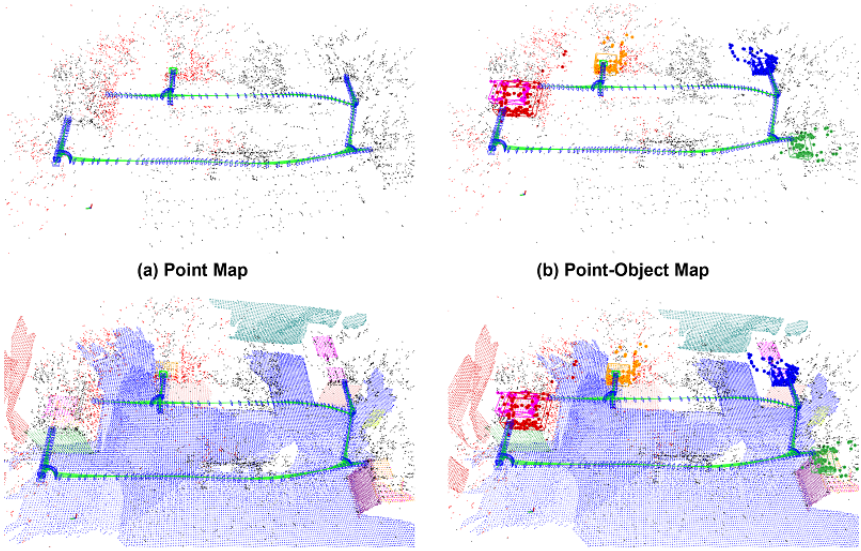


Figure 3.25: Comparison of the mapping results using different SLAM systems on the logistics environment.

3.5 Chapter Conclusion

In this chapter, the introduction of 3D objects is explored to benefit scene understanding and localization. To answer this question, a visual SLAM system called point-plane-object SLAM is proposed. The whole system is built on ORB-SLAM2, in front-end, it detects and tracks more features in every key-frame, such as geometric planes and semantic objects. In terms of data association, new strategies are designed to ensure a robust system. For back-end optimization, these landmarks are integrated into a unified bundle adjustment framework to jointly optimize camera poses. Besides, the object-based loop detection module is considered under different viewpoints. Finally, a sparse semantic map with different landmarks can be built, including feature points, planes, and objects.

The proposed method is evaluated on indoor and logistics environments. Results show that the proposed point-plane-object SLAM system achieves camera pose estimation and is able to build a sparse semantic map with semantic landmarks. Compared to other visual SLAM systems, the camera localization accuracy is improved.

Since ORB feature-based initialization may fail in some cases, future work should focus on a better initialization system with planes or other landmarks. Besides, the current method requires objects to be fully visible and ignores objects that are only partially observed in the image. It is worth exploring if these partially observed objects could also provide additional information for the SLAM system.

4 Efficient Object-Level Mapping with RGB-D Cameras

In this chapter, a semantic map is incrementally built with RGB-D cameras. Different from the sparse map for localization in Chapter 3, the reconstructed map here is dense and aims to benefit navigation and grasping. Section 4.1 introduces the background of the mapping system. Section 4.2 reviews related work on different maps, including grid maps, feature-based maps, point cloud maps, and voxel-based maps. Specially, objects are introduced to these maps. Section 4.3 presents the proposed object-level mapping system, where new strategies are employed to track and update objects across multiple frames and incorporate them into a voxel-based map. Section 4.4 designs the experiments on indoor and logistics environments to evaluate the performance of the proposed method, the results show that the method can efficiently build an object-level volumetric map while reducing computational costs. Finally. Section 4.5 closes the chapter. This chapter's work is an extension to object detection and robot localization in Chapter 2 and Chapter 3.

4.1 Introduction

Semantic mapping aims to estimate the geometry of an environment and simultaneously attach a semantic label to the elements that are reconstructed in the map. With the aid of RGB-D cameras, the mobile robot can perceive object information from the surrounding space, generate an up-to-date map with semantics, and design a collision-free path to achieve navigation.

Considering real-world applications, three fundamental challenges are addressed in semantic mapping: object detection, geometric reconstruction, and computation efficiency. Chapter 2 provides a comprehensive discussion on various object detection methods. In terms of 3D reconstruction, a dense map with occupancy and object information is required for autonomous navigation and other intelligent tasks. For efficiency, although there is a lot of research work developing semantic mapping systems with RGB-D cameras, they suffer from critical real-time issues due to the heavy processing components, such as pixel-wise object segmentation and point cloud processing. Current GPU technology can accelerate the computation process, but it is not always available and practical for some industrial projects due to cost, compatibility, and space constraints.

In this chapter, a CPU-based semantic mapping system is presented to incrementally build an object-level map with a localized RGB-D camera. For each frame, the detected objects are represented with the point cloud model. Then, an object association strategy considering geometric and semantic descriptors is proposed to match the detected objects in the current frame to existing mapped objects, where the detected objects will be either merged or introduced in the map. Finally, the object instances will be integrated into a voxel-based mapping system to incrementally reconstruct a global object-level volumetric map. Experiments on publicly available indoor datasets show that the proposed system achieves a comparable semantic mapping performance while reducing the computational cost. Furthermore, the system is evaluated within a logistics robotic platform to demonstrate the use case in real-world applications. In a nutshell, the contributions can be summarized as follows:

- An object association strategy based on geometric and semantic descriptors to track and update object information.
- An CPU-based object-level mapping algorithm, where the objects are introduced into a 3D volumetric map.
- Experiments on a public open-source dataset and a real-world robotic platform to evaluate the performance of the proposed system.

Table 4.1: Literature review of semantic mapping systems.

	Sensor	Map Type	Objects	Features
Grisetti et al. (2007)	laser scanner	grid map		2D grid
Leigh et al. (2015)	laser scanner	grid map	labelled points	hard-code features
Himstedt and Maehle (2017)	RGB-D	grid map	visual objects	top-to-down projection
Pang et al. (2019)	laser scanner, RGB	grid map	labelled points	back-projection
Zaenker et al. (2020)	laser scanner, RGB-D	grid map	polygon	multiple layers
Sivananda et al. (2022)	laser scanner, RGB-D	grid map	object model	multiple layers
Newcombe et al. (2011)	RGB-D	point cloud map		dense map
Pham et al. (2015)	RGB-D	point cloud map	object point	features and contextual information
Sünderhauf et al. (2017)	RGB-D	point cloud map	object point	object segmentation
McCormac et al. (2017)	RGB-D	point cloud map	object point	object segmentation
Hornung et al. (2013)	RGB-D	voxel map		3D occupancy octomap
Oleynikova et al. (2017)	RGB-D	voxel map		3D TSDF map
Rosinol et al. (2020)	RGB-D	voxel map	object mesh	semantic segmentation
Pham et al. (2019)	RGB-D	voxel map	object voxel	super-voxel clustering
Grinvald et al. (2019)	RGB-D	voxel map	object voxel	volumetric object-instance map
Li et al. (2020b)	RGB-D	voxel map	object voxel	Gaussian mixture model
Mascaro et al. (2022)	RGB-D	voxel map	object voxel	label fusion

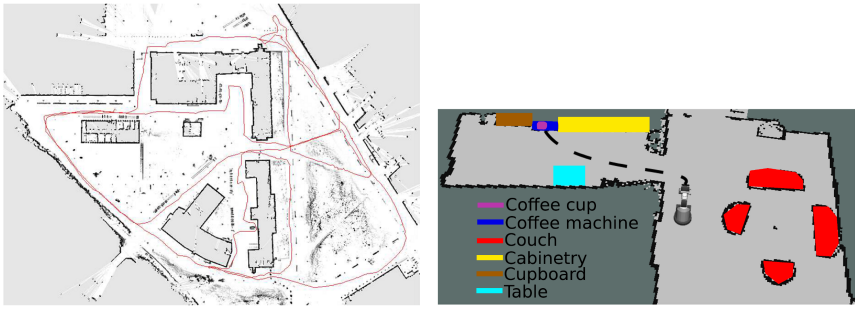
4.2 Literature Review

In 3D reconstruction, mapping refers to the process of converting sensor information into a representation of the environment. Here four maps with different functions are discussed. An occupancy map is a typical indoor map that is used for planar navigation. It decomposes the space into fixed-size grids, and each grid cell indicates whether this area is occupied or not. A feature-based map represents the environment as different landmarks, including feature points, geometry planes, semantic objects, etc. This map is flexible to update, thus mainly used for scene understanding and localization. A point cloud map is a dense map that reconstructs the world with hundreds of points, which shows the detail of the environment and is intuitive for visualization. Finally, a voxel-based map models the environment with cubic volumes of equal size and discretizes the mapped area with explicitly free space. Since it contains occupancy information, this map provides a solution for 3D navigation. In the following, these maps will be reviewed, and the focus will be on introducing semantics to them.

4.2.1 2D Grid Map with Objects

A 2D occupancy grid map is a common solution for indoor planar navigation, where the environment is represented as a fixed, regular grid, and each cell indicates the probability that this area can be traversed or not. The 2D map can be generated automatically with laser scan data using laser-based SLAM algorithms, such as Gmapping (Grisetti et al. 2007), Hector SLAM (Kohlbrecher et al. 2011), and Cartographer (Hess et al. 2016). Although occupancy maps are convenient for navigation, the lack of semantic information limits their suitability for high-level tasks. To address this limitation, additional information can be incorporated into the map to improve environmental understanding and navigation flexibility.

Introducing high-level elements to the map is under investigation. Some researchers explore segmenting objects directly from the laser scan data. Leigh



(a) Grid map (Grisetti et al. 2007)

(b) Grid map with objects (Dengler et al. 2021)

Figure 4.1: Example of 2D grid maps with objects. They usually represent the environment as a fixed, regular grid and label the objects with different colors. The figures come from the original work.

et al. (2015) separated a scan into several connected groups and extracted hand-crafted features to label these points. By assigning different labels to the laser points, a meaningful map can be created. However, the 2D laser scanner limits the detection on a 2D plane, and can not detect objects at different heights, such as a book on the table. In contrast, Himstedt and Maehle (2017) presented to utilize only one RGB-D camera for semantic mapping. They detected objects from images, converted the labelled point cloud into annotated scan data with top-to-down projection, and integrated them into a probabilistic SLAM framework to generate a semantic map.

While cameras provide an efficient way to detect objects, laser scanners show a better performance in distance measurement and accurate map creation. A multi-sensor system can be implemented to build a hyper-map. For example, based on a pre-built 2D occupancy grid map with laser scan data, Sivananda et al. (2022) detected objects from an RGB image, searched for a corresponding model in the library, and added the model to the existing map. Pang et al. (2019) attempted to segment semantic laser points by incorporating image data. They achieved this by back-projecting laser scan data onto an RGB image. By obtaining the object bounding box in the image, they were able to extract corresponding laser points and utilize them for mapping purposes. Zaenker et al. (2020) introduced

the hyper-map framework, which involves the construction of multiple map layers individually, including the occupancy layer and semantic polygon layer. This framework effectively manages the different layers and enables their utilization for various purposes.

4.2.2 Feature-based Map with Objects

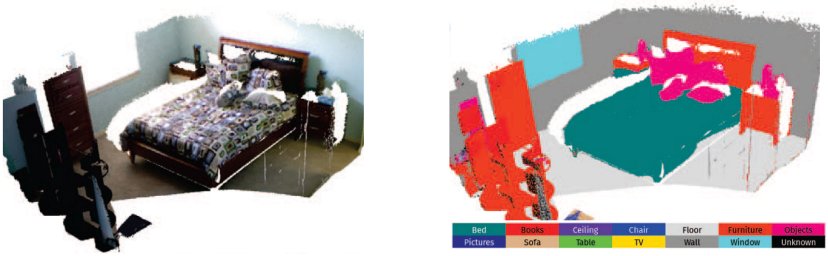
The environment can be reconstructed by a list of landmarks, such as feature points, geometric planes, semantic objects, or other essential components. Geometric landmarks provide a concise representation of the scene, while semantic features offer long-term stability and resilience against perspective changes. A complete review of the feature-based map can be found in Section 3.2.

A feature-based map is sparse and easy to update, making it a good option for localization and scene understanding. However, due to the absence of distance information, it cannot be directly employed for path planning purposes.

4.2.3 Point Cloud Map with Objects

By collecting 3D points, the environment can be represented as a point cloud map, where each point can store color, position, and other information. This map can be generated by various sensors, including LiDAR and RGB-D cameras. Point cloud maps are widely used in computer vision and robotics because they provide a high level of detail, precision, and intuitive visualization.

Newcombe et al. (2011) presented a detailed method to permit real-time, dense reconstruction of complex room-sized scenes using a handheld RGB-D camera. The high-quality map can enable a full physical predictive interaction between virtual and real scenes. Adding objects to a point cloud map can be regarded as assigning object labels to each point, indicating whether it belongs to a specific category or not. Pham et al. (2015) addressed the problem by introducing a novel higher-order model for semantic 3D indoor scene labelling, which allows



(a) Point cloud map (McCormac et al. 2017)

(b) Point cloud map with objects (McCormac et al. 2017)

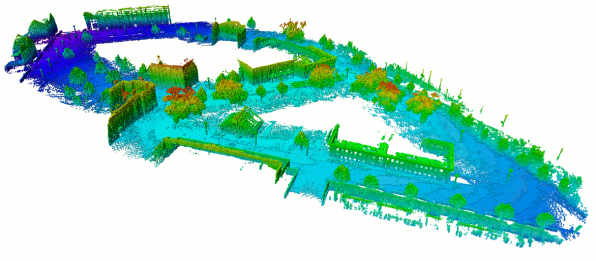
Figure 4.2: Example of point cloud maps with objects. They usually represent the environment as hundreds of points and label the object with different colors. The figures come from the original work.

the incorporation of features and contextual information for better performance. Sünderhauf et al. (2017) proposed to create a meaningful map by maintaining individual objects as the key entities in the map, McCormac et al. (2017) utilized a pre-trained convolutional neural network to process RGB-D pair with a per-pixel class probability distribution and fused the instance information into a coherent 3D semantic map with a Bayesian update strategy.

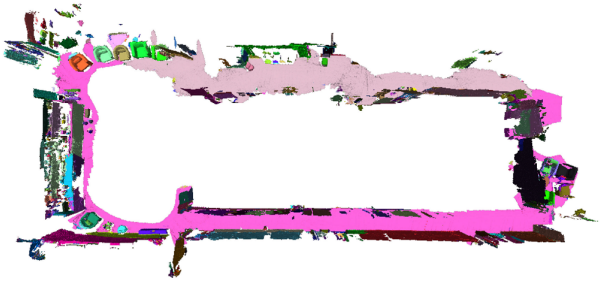
Although a point cloud map can model the environment with high precision, it can not represent either free space or unknown areas. Therefore, it is not efficient for path planning. Furthermore, the computational complexity becomes problematic due to the extensive number of points in the map, leading to significant demand for memory and computation resources.

4.2.4 Voxel-based Map with Objects

A voxel-based map is a 3D map representation that uses voxels, or cubic volumes, to store environmental information. Each voxel can be either occupied or free, corresponding to its occupancy probability. Octomap (Hornung et al. 2013) is a typical 3D voxel-based map that uses a tree-based representation to efficiently



(a) Octomap. (Hornung et al. 2013)



(b) TSDF(voxel) map with objects. (Grinvald et al. 2019)

Figure 4.3: Example of voxel maps with objects. They usually represent the environment as voxels or cubic volumes, and label the objects with different colors. The figures come from the original work.

process and update large-scale 3D information. It performs a probabilistic occupancy estimation, allowing for uncertainty and sensor noise. On the basis, Liu et al. (2019b) extended the octree to include object information and create an object-aware semantic map for the indoor scene. The main drawback of octomap is that the maximum size of the map must be known a priori and cannot be dynamically changed.

A Truncated Signed Distance Field (TSDF) map is another type of voxel map, it represents the environment using only signed distance information rather than occupancy probability. Each voxel saves the signed distance to the closest surface points, indicating the points are outside or inside the surface. This information is valuable for surface reconstruction and collision detection. Voxblox (Oleynikova et al. 2017) is one of the CPU-based mapping systems that can densely reconstruct

volumetric TSDF maps in unexplored environments, providing valuable free and occupancy space information to guarantee navigation safety. Adding semantics to these maps, Rosinol et al. (2020) presented Kimera-Semantics to create a semantic map, where they annotated semantics with 2D pixel-wise segmentation and built a global 3D mesh using voxel-based (TSDF) approach.

Previous work has addressed semantic mapping of the whole environment, while other object-oriented approaches focus on identifying and reconstructing individual objects. Pham et al. (2019) designed a higher-order Conditional Random Field (CRF) to infer optimal segmentation labels and employed an efficient super-voxel clustering method for object segmentation in 3D indoor scenes. Grinvald et al. (2019) presented a combined geometric-semantic scheme to incrementally build a volumetric object-centric map, which retrieves both recognized scene objects as well as previously unobserved elements. Inspired by this work, Li et al. (2020b) represented object-instance as a Gaussian mixture model considering the projection relationship between voxel and pixel. Mascaro et al. (2022) emphasized the data association module and employed a label diffusion scheme to regularize the final instance segmentation. The above methods can achieve high performance in 3D object segmentation accuracy while building a semantic map. One of the common issues of the above approaches is that they rely on 2D semantic masks, thus require high computational costs, and can not run on CPU-based robotic platforms.

4.3 Method

The proposed object-level semantic mapping system is illustrated in Figure 4.4, which takes RGB-D sequences as input and incrementally builds a volumetric map enriched with object instances. To achieve these functions, the RGB-D sequences are initially processed by a camera pose tracking framework (see Section 4.3.1). Then, an object detection method is employed to segment 3D semantic objects from each frame (see Section 4.3.2). After that, these frame-based detected objects are matched to globally mapped objects via an object association strategy, which uses geometric and semantic descriptors to track and update object information (see Section 4.3.3). Finally, the associated objects are incorporated into a TSDF volumetric mapping framework to generate an object-level dense map (see Section 4.3.4).

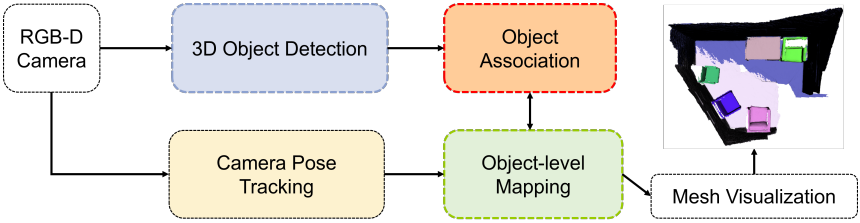


Figure 4.4: Overview of the proposed semantic mapping system. It takes RGB-D sequences as input to build a volumetric object-oriented map. The input is processed by camera pose tracking and object instance segmentation processes to get camera localization and object information, then, an object association strategy is adopted to update objects across multiple frames, and finally, all objects are integrated into a 3D volumetric map.

4.3.1 Camera Pose Tracking

The proposed mapping system requires camera poses when robots move in an unknown scene. The movements can be estimated by wheel encoder, feature tracking (Mur-Artal and Tardós 2017), and laser scan matching (Grisetti et al. 2007), depending on which sensor is available. As in Chapter 3, many SLAM

systems emphasize how to accurately estimate camera poses with RGB-D sequences. In this chapter, the camera pose tracking is assumed as solved and the focus is on the mapping process.

4.3.2 Object Detection

For each RGB-D frame, the geometry method (see Section 2.3.2) is chosen to detect 3D objects, where the semantic and geometric information are included, such as class, 2D bounding box, 3D cuboid, associated feature points, etc. It is worth noting that other methods can also be implemented, more details can be found in Chapter 2.

4.3.3 Object Association

Since the frame-wise segmentation processes each image independently, an object association module is proposed to determine correspondences among multiple frames and incrementally update the object in the global map. It should be noted that the object association strategy employed in this chapter differs from that in Chapter 3, especially with respect to the update strategy. Chapter 3 uses a parametric form to accelerate the updating process, whereas in this chapter, the point cloud model is used to incrementally maintain a dense reconstruction.

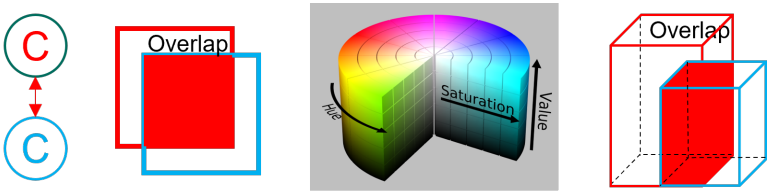


Figure 4.5: Proposed data association strategy for semantic mapping. From left to right are: class category; 2D IoU; HSV(which stands for Hue Saturation Value); 3D IoU.

For each frame detected object A and global mapped object B, geometric and semantic descriptors are extracted to achieve an accurate matching. Firstly, their class and frame ID are checked, where a frame closeness is defined $\eta(A, B)$ as:

$$\eta(A, B) = |Frame\_ID(A) - Frame\_ID(B)| \tag{4.1}$$

where $Frame\_ID$ are represented as numbers and $|.|$ measure the difference.

Secondly, the similarity in image space is checked. Since 2D bounding box and color information can be obtained. The 2D bounding box can be used to calculate 2D IoU (Intersection over Union), while the color information inside the 2D bounding box can be extracted to match objects. In this work, the images of the objects are converted to the HSV (Hue Saturation Value) color space, then, the color histograms of the two images are calculated to generate the normalized feature vectors, finally, the appearance similarity $Sim(A, B)$ between the two objects can be computed using these vectors. The IoU_{2D} and $Sim(A, B)$ are calculated as:

$$IoU_{2D} = \frac{BBOX_{2D}(A) \cap BBOX_{2D}(B)}{BBOX_{2D}(A) \cup BBOX_{2D}(B)} \tag{4.2}$$

$$Sim(A, B) = \frac{HSV(A) \times HSV(B)}{\sqrt{HSV(A)^2 + HSV(B)^2}} \tag{4.3}$$

Thirdly, after obtaining the cuboid representation of each object, their spatial relationship is checked by calculating the 3D IoU as follows:

$$IoU_{3D} = \frac{BBOX_{3D}(A) \cap BBOX_{3D}(B)}{BBOX_{3D}(A) \cup BBOX_{3D}(B)} \tag{4.4}$$

To sum up, two objects are considered as an associated pair if they satisfy the following constraints: their class should be the same, $\eta(A, B) < 10$, $IoU_{2D} > 0.5$, $Sim(A, B) > 0.3$, $IoU_{3D} > 0.5$. These values guarantee a stable performance in the experiments. When applied to other environments, they may be different.

For every frame-detected object, If there are no existing mapped objects associated, it will be assigned with a $Mapped\_ID$ and added to the global map. A mapped object should contain the following parameters: $< Frame\_ID$, $Mapped\_ID$, $Class$, $bbox$, $\mathbf{T}$, $\mathbf{R}$, $\mathbf{D}$, HSV, $3D\ points, ... >$. In contrast, when one frame-detected object is matched to a mapped object, it should be merged into this mapped object and updated the information. In this case, the semantic information ($Frame\_ID$, $Class$, $bbox$, and HSV) will be replaced by new information, while the 3D points are accumulated and the cuboid parameters ($\mathbf{T}$, $\mathbf{R}$, $\mathbf{D}$) of the 3D cuboid will be computed again as described in Section 2.3.2.

4.3.4 Object Mapping

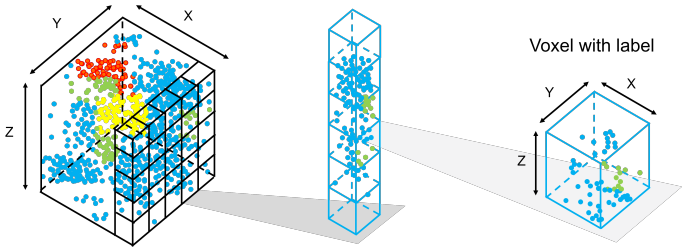


Figure 4.6: Proposed voxel-based mapping process. It converts the labelled 3D point cloud (blue, red, and yellow) into a voxel-based map, where each voxel is extended to store the object label, and this information will be incrementally updated by counting object labels inside the voxel (blue).

The object-level map is expected to not only contain object information to better understand the scenario, but also provide valuable information for navigation. In this case, Voxblox (Oleynikova et al. 2017) is opted as the foundational framework to map the environment using TSDF information. This decision stems from the fact that each voxel incorporates signed distance data, which directly enhances navigation safety. Additionally, Voxblox is a real-time solution designed for CPU usage, making it particularly well-suited for integration with robotic platforms in real-world scenarios.

Raycasting (Curless and Levoy 1996) technology is widely used in many voxel mapping systems to project an RGB-D image onto the voxel grid, which casts a ray from the camera optical center to the center of each observed point, and updates all voxels from the center to the truncation distance behind the points. On this basis, Voxblox uses a grouped raycasting approach to significantly speed up the mapping process without losing much accuracy. They projected each point to the voxel grid, grouped it with other points mapping to the same voxel and the mean distance of all grouped points, by this way the raycasting only needs to perform once. This leads to a similar reconstruction result while enabling a real-time solution on the CPU platform.

The mapping method is augmented with object information, where every voxel is extended to store object labels. As illustrated in Fig. 4.6, in every frame, the 3D points are segmented as object points and "background" points, where the object points are assigned object labels after the 3D segmentation, while other points are labelled with “0”. During the voxelization process, in each voxel v, the number of points with label li can be accumulated as:

$$\psi(v, li) \leftarrow \psi(v, li) + 1. \tag{4.5}$$

After all points are converted into voxel, the voxel label $L(v)$ will be updated by the object label that has the maximum count as:

$$L(v) = \underset{i}{\operatorname{argmax}} \psi(v, li), \tag{4.6}$$

In the process of converting the labelled 3D points into voxels, the object detection and association modules ensure consistency across different frames to provide stable object labels.

4.4 Experiments

The performance of the proposed system is evaluated on indoor environments from SceneNN dataset (Hua et al. 2016), which features RGB-D scans of different indoor scenes, including offices, bedrooms, and kitchens. This dataset provides the ground truth with object instances, which makes it suitable for comparing the reconstruction results with object-level mapping approaches.

Figure 4.7: SceneNN dataset. This dataset provides several RGB-D sequences of different indoor scenes with object instance. The figure comes from the official website: https://hkust-vgd.github.io/scenenn/.

To run the experiments, a ThinkPad laptop (i7-8565U 1.80 GHz CPU, 16 GB RAM, no GPU, Ubuntu 18.04) is used. Due to hardware constraints, YOLOv2 (Redmon and Farhadi 2017) is chosen as the object detector with general pre-trained weights downloaded from the official website. The mapping framework, as well as mesh visualization tools, comes from (Grinvald et al. 2019). All components, including object detection, association, and mapping are implemented in C++. To show the efficiency, the proposed system is also transferred to a GPU platform and the runtime performance is reported. Furthermore, the applicability of the method are demonstrated on a real robotic platform with RGB-D sensors, where the mobile robot is driven to map an unknown intra-logistics environment.

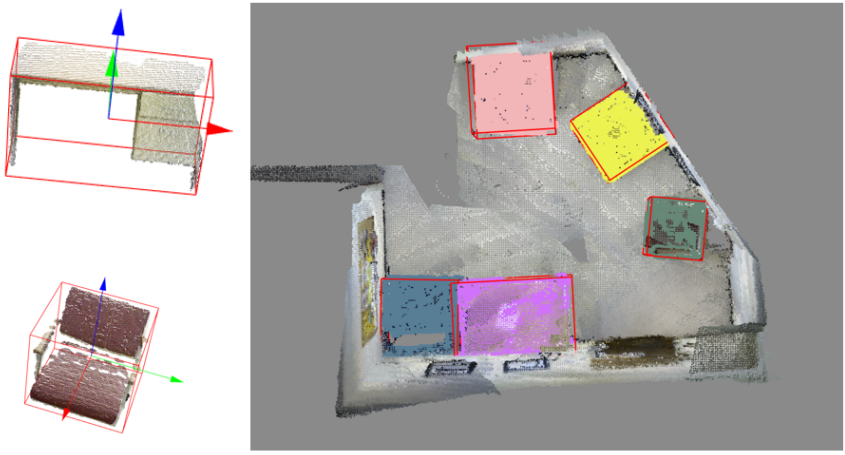
4.4.1 Experiments on Indoor Environments

1) Qualitative Results. one example of the object-level mapping results is first visualized on sequence 011 of the SceneNN dataset. As shown in Figure 4.8(**a**), the left side shows the point model of a table and a chair from a single frame, while the right side shows the whole point cloud map with objects. The corresponding voxel models of objects and TSDF map are also displayed in Figure 4.8(**b**).

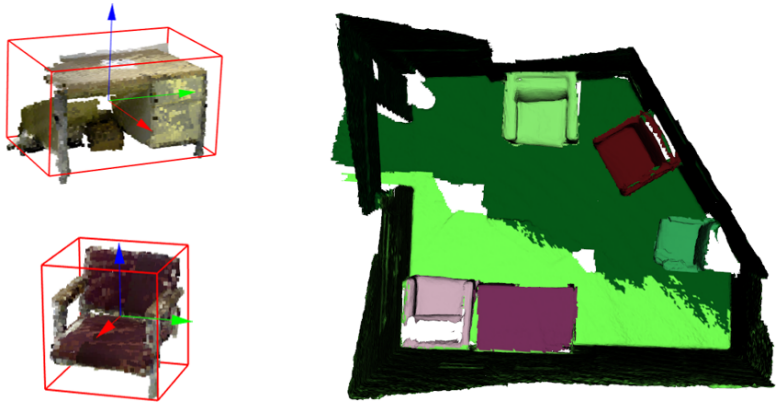
More mapping results can be found in Figure 4.9, where the second column is the input RGB image with the 2D bounding box, and the third column shows the object points with cuboid representation. Although the objects are partially observed, they will be associated and incrementally updated with multi-view optimization. Finally, the point cloud maps and volumetric maps augmented with individual objects are shown in the last two columns. It can be seen that a dense map with object information can be obtained.

2) Quantitative Results. During the mapping process, the TSDF map has a voxel size of $2cm$. It is difficult to quantify the difference between the reconstructed map and the ground truth. Instead, a possible way to evaluate the accuracy of the map is by comparing the object positions within it. Following the evaluation procedure introduced by (Grinvald et al. 2019), the object mapping algorithm is assessed on 10 indoor sequences from SceneNN dataset (Hua et al. 2016), 9 object categories (i.e., bed, $chair$, $sofa$, $table$, $books$, $refrigerator$, $television$, $toilet$, and bag) are considered for comparison with other research work. For each sequence, the per-class Average Precision (AP) score is computed using the 3D Intersection over Union (IoU) threshold of 0.5 over segmented objects and ground truth. The mean Average Precision (mAP) of each sequence is directly calculated by averaging the per-class AP scores.

Table 4.2 and Table 4.3 show the 3D IoU and mAP results of 10 sequences. Besides, Table 4.4 illustrates the comparison of mAP with other object-level mapping systems (Oleynikova et al. 2017, Grinvald et al. 2019, Mascaro et al. 2022), where the data is cited from the original paper. While other methods use pixel-wise masks for object segmentation, the object-wise detection and cluster



(a) Point cloud map with objects.



(b) Mesh map with objects.

Figure 4.8: Mapping results on sequence 011 of SceneNN dataset. The left shows the objects model, while the whole map is on the right. It is worth noting that voxel-based maps are visualized as mesh maps.

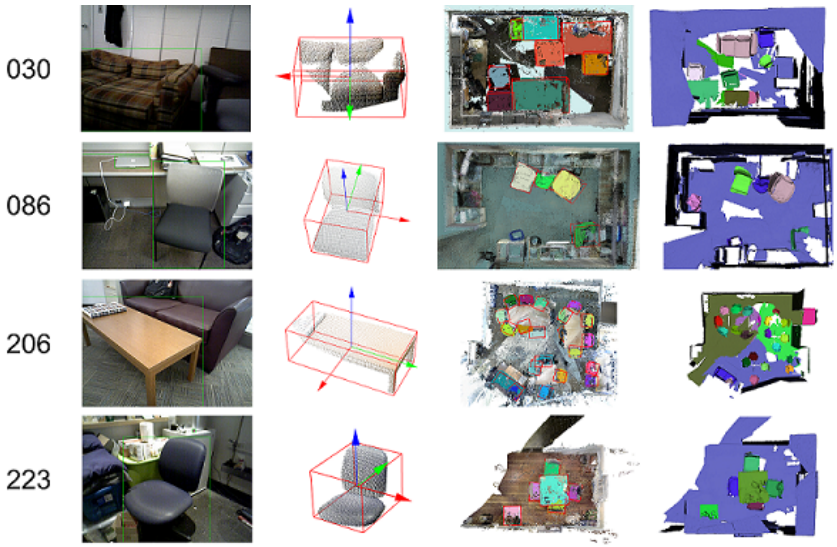


Figure 4.9: Example of detection and mapping results on SceneNN dataset. The second column is the input RGB image with the 2D bounding box, the third row shows the object points with cuboid representation, and the last two columns present the point cloud maps and volumetric maps augmented with individual objects

method can also achieve good segmentation accuracy. These tables demonstrate that the proposed approach outperforms other baselines on 6 of the 10 evaluated sequences, and triggers a significant increase in the achieved 3D segmentation accuracy. However, it is worth pointing out that the reported mAP values are computed over a smaller set of classes.

Focusing on specific categories, the segmentation performance varies in different objects: $sofas$ and $chairs$ achieve a better segmentation performance due to their size and multiple observations. As shown in the last row of Figure 4.9, although the $chair$ is partially observed, the object association strategy, which utilizes geometric and semantic descriptors, provides robust object tracking and benefits object refinement. Besides, it is observed that object segmentation with bounding boxes might lead to over-segmentation or overlap problems, in the third row of Figure 4.9, the detected bounding boxes of $table$ and $sofa$ are overlapped, in this

Table 4.2: Evaluation of 3D object detection accuracy (IoU_{3D}) of 10 sequences from SceneNN dataset. "-" means this object type doesn't exist in this sequence.

Sequence ID	Bed	Chair	Sofa	Table	Books	Refrigerator	Television	Toilet	Bag	Average
011	-	70.2	70.2	86.3	-	-	-	-	-	78
016	51.3	-	71.9	-	-	-	-	-	-	41.4
030	-	57.6	80.4	85.7	0	-	-	-	-	57.4
061	-	74.5	62.7	95.1	-	-	-	-	-	77.4
078	-	45.9	-	0	0	67.8	-	-	-	13.9
086	-	58.2	-	0	0	-	-	-	53.8	56.0
096	63.1	60.9	-	0	0	-	32.3	-	0	31.3
206	-	56.5	23.3	65.5	-	-	-	-	29.6	43.7
223	-	63.7	-	69.2	-	-	-	-	-	66.5
255	-	-	-	-	-	55.8	-	-	-	55.8

case, the method first segments the $table$ as it has a convex shape with surface planes and is easy to estimate from the environment. Then, inside the $sofa$ bounding box, it can remove the points that belong to $table$ to prevent incorrect detection. However, there are some disadvantages of the proposed system, as shown in Table 4.2, it is hard to distinguish small objects like $books$ from the cluttered background because there are not enough points remaining after the exclusion of outliers.

Table 4.3: Evaluation of 3D object detection accuracy (mAP) on 10 sequences from the SceneNN dataset.

Sequence ID	Bed	Chair	Sofa	Table	Books	Refrigerator	Television	Toilet	Bag	Avgerage
011	-	100	100	100	-	-	-	-	-	100
016	100	-	100	-	-	-	-	-	-	66.7
030	-	72	100	66.7	0	-	-	-	-	59.7
061	-	62.5	100	33.3	-	-	-	-	-	65.3
078	-	50	-	0	0	100	-	-	-	37.5
086	-	75	-	0	0	-	-	-	50	31.3
096	100	100	-	0	0	-	0	-	0	33.3
206	-	41	0	40	-	-	-	-	0	20.3
223	-	100	-	50	-	-	-	-	-	75
255	-	-	-	-	-	100	-	-	-	100

Table 4.4: Comparison of the 3D object segmentation accuracy (mAP) among these methods, (Pham et al. 2019), (Grinvald et al. 2019), and (Li et al. 2020b). The results come from the original research work, Best results on each sequence are highlighted in bold.

Method	011	016	030	061	078	086	096	206	223	225	Average
Pham et al. (2019)	52.1	34.2	56.8	59.1	34.9	35.0	16.5	41.7	40.9	48.6	43.0
Grinvald et al. (2019)	75.0	33.3	56.1	**66.7**	45.2	20.0	29.2	**79.6**	43.6	75.0	54.4
Li et al. (2020b)	78.6	25.0	58.6	46.6	**69.8**	**47.2**	26.6	78.0	45.8	75.0	55.1
Ours	**100**	**66.7**	**59.7**	65.3	37.5	31.3	**33.3**	20.3	**75**	**100**	**58.9**

3) Runtime Performance. Another advantage of the proposed system is the runtime performance, which is analyzed and compared with other state-of-the-art systems. Table 4.5 shows the evaluation of the execution times of the individual model of the proposed pipeline averaged over 10 evaluated sequences in SceneNN (Hua et al. 2016) dataset. The object detection module running by YOLOv2(Redmon and Farhadi 2017) takes 725ms to detect 2D bounding boxes in each frame, which is the most time-consuming process, it can be accelerated when transferred to a GPU platform (NVIDIA RTX 3050TI GPU). When compared to other systems, as shown in Table 4.6, the system achieves a speed of 1 Hz in CPU and 10 Hz in GPU. Li et al. (2020b) also uses YOLOv2 as the 2D object detector and reaches a comparative speed, while other methods employ Mask R-CNN (He et al. 2017) to generate pixel-wise segmentation and show slower performance. The computational time can be substantially reduced in two fields, one is to use a fast and stable object detector, followed by a robust outlier exclusion method to segment 3D objects from RGB-D images. The other is to exploit a voxel-based mapping and update scheme that can run in real time on the CPU. Most importantly, the system can be extended to a real robotic platform without GPU requirements, which provides a convenient and cheap solution for real-world applications.

Table 4.5: Average execution time of each processing module in the proposed system. Note that a separate thread is created for the object mapping module, which does not affect the frame rate of the whole system.

Module	Runtime-CPU (mSec)	Runtime-GPU(mSec)
Object Detection	725	32
Object Segmentation	17.75	15
Object Association	5.22	5
Object Mapping	299	50
Total	1046.97	102

Table 4.6: Comparison of runtime performance on SceneNN dataset.

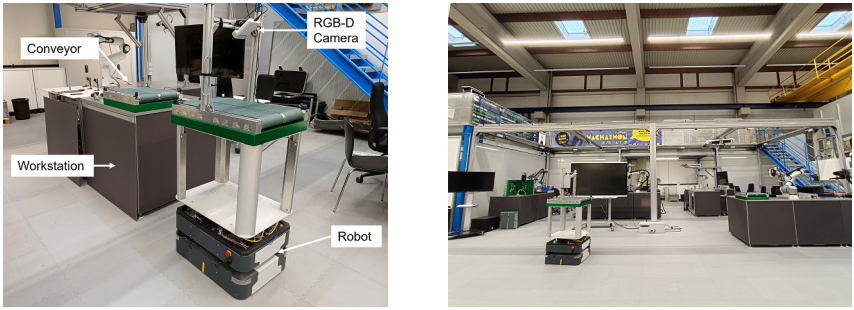
Method	CPU/GPU	Frequency	map	FPS
Pham et al. (2019)	GPU	every frame	object-oriented	1 Hz
Grinvald et al. (2019)	GPU	every frame	object-oriented	1 Hz
Li et al. (2020b)	GPU	every frame	object-oriented	10.8 Hz
Ours-CPU	CPU	every frame	object-oriented	1 Hz
Ours-GPU	GPU	every frame	object-oriented	10 Hz

4.4.2 Experiments on Logistics Environments

The system is also evaluated within the AgiProbot project (Klein et al. 2021), which is an intra-logistics system for an agile remanufacturing product system built at Karlsruhe Institute of Technology (KIT). In this project, the mobile robots are assigned to transport and transfer items among different workstations. Specifically, as shown in Fig. 4.10, the reconstructed map needs to not only contain occupancy information for global navigation, but also integrate object information for precise docking. In this case, the object-level semantic mapping system provides a suitable solution to build a dense map with semantic objects.

To better accomplish these tasks, a 2D SICK laser scanner and a Microsoft Azure Kinect camera are mounted. The SICK laser scanner is horizontally installed at a height of 0.3 m and in the northwest corner of the vehicle. It can measure a maximum distance as $30m$ with $270°$ scanning angle and publish the scan data up to 30 Hz. The Microsoft Azure Kinect camera is mounted at a height of 1.4 m on the top of the vehicle center, facing downwards at $45°$. It captures RGB and depth images in 720P resolution (1280x720) with $90° \times 60°$ field of view (FOV) and publishes images up to 15 Hz. Calibration[1] between the laser scanner and the camera is solved by (Zhang and Pless 2004).

[1] https://github.com/MegviiRobot/CamLaserCalibraTool



(a) The robotic platform. **(b)** The AgiProbot environment.

Figure 4.10: The robotic platform in AgiProbot project.

To generate an object-level map, the mobile robot drives around the whole environment ($6 * 12m$) to record laser scan data and RGB-D images. The laser scan data are used for the camera (robot) pose tracking, while the RGB-D sequences are fed to the object-level mapping framework to build a semantic map. Since no GPU is available on board, the frame rate is set as 1 Hz, any frames that exceed the processing abilities of the system are discarded and not used to reconstruct the object-level map of the scene.

Camera Pose Tracking. The dependency of robot localization is solved by ICP registration (Censi 2008) on laser scan data, which achieves a good result in robot pose tracking. On this basis, a point cloud map is generated in Figure 4.11. Compared to the real environment, the point cloud map provides a detailed description, which identifies that the camera pose tracking module functions well. The object from the point cloud map can also be manually annotated as ground truth to evaluate object detection results.

Object Detection. For object detection, 100 images containing $conveyors$ are captured and manually annotated to train the YOLO neural network (Redmon and Farhadi 2017), after that, the pre-trained weights are used to detect $conveyor$ with bounding boxes in every frame. During the mapping process, object detections are removed when the robot is rotating because they may introduce inaccurate

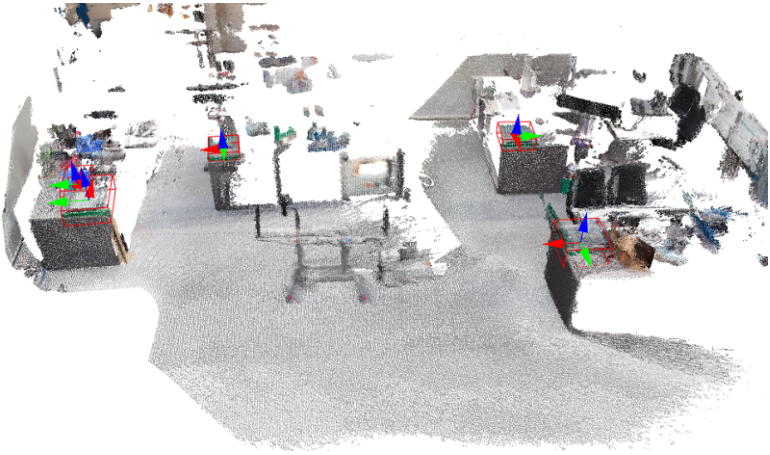


Figure 4.11: The point cloud map of the logistics environment. It tracks robot poses with laser scan data and generates a point cloud map with the RGB-D camera, where the ground truth of objects is labelled.

Table 4.7: Evaluation of object detection on the logistics environment

Object ID	IoU_{3D}	$E_{trans}(m)$	$E_{rot}(^\circ)$
1	0.7446	0.058	3.4
2	0.7140	0.060	0.9
3	0.8061	0.029	1.6
4	0.9056	0.035	1.0
Average	0.7925	0.045	1.7

measurements to the map. Additionally, to improve accuracy, detections that are far away from the camera due to the detection range are removed.

After tracking and updating object information, 3D objects can be segmented and the results are shown in Table 4.7. To evaluate the precision of the methodology, 3D IoU (IOU_{3D}), translation error (E_{trans}), and rotation error (E_{rot}) are employed as metrics. The translation error is determined by the centroid distance in

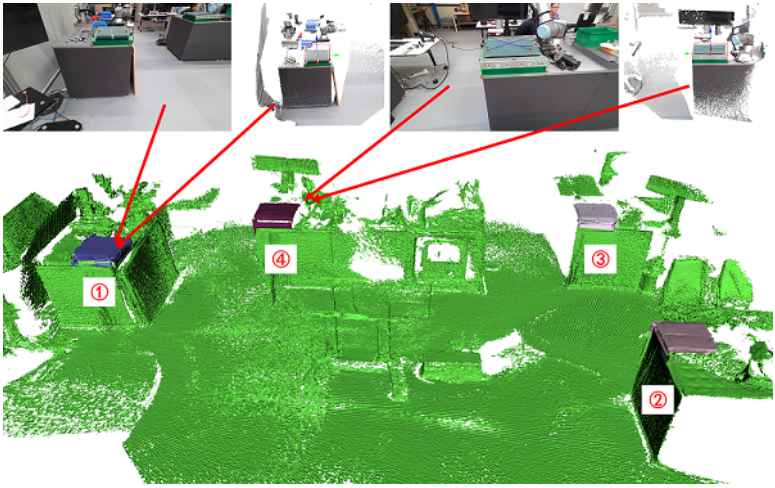


Figure 4.12: Mapping results on the logistics environment. The figures on the top show the frame-detected objects in the 2D image and 3D space, while the reconstructed map is visualized below, where the conveyors are assigned with different colors while the background points are green.

3D space, while the rotation error is calculated by the yaw of the object in the top viewpoint. As shown in the table, the object IoU is an average of 0.79, and the translation error and rotation error are $0.045m$ and $1.7°$ respectively. Two important factors may influence the results. One is sensor noise. It can be observed that when the robot is rotating, the 2D object detections and point cloud measurements are inaccurate and will deteriorate the segmentation result. Therefore, these bad detections are removed. The other is the object updating strategy. Object points are simply accumulated, and object geometry is computed; it is challenging to remove outliers. A better idea is to calculate and update the probabilities of all corresponding 3D points.

Object Mapping. The results for the reconstructed map with a voxel resolution of $2cm$ are shown in Figure 4.12, where $conveyors$ are marked with different colors while the background is green. It is observed that the proposed method can integrate incoming RGB-D images into the map volume, providing a comprehensive representation of the surface geometry for individual objects. This

volumetric map contains object information to better understand the scene, and the additional free space information is relevant to safe planning for autonomous navigation. Although the system operates at only 1 Hz with CPU, it validates the online framework and shows its benefit for real-world applications. A demo video [2] is uploaded to illustrate the whole process of incrementally reconstructing the semantic object-level map of the scene.

4.5 Chapter Conclusion

In this chapter, a CPU-based object-level semantic mapping system is presented, which takes RGB-D sequences as input to build a volumetric object-oriented map. Firstly, the RGB-D sequences are processed with an object detection module to segment object points from a single frame. Then, a data association strategy with geometric and semantic descriptors is designed to track and update object information. Finally, the partial segmentation information is incrementally fused into a global map and results in an object-level volumetric map, which can be further used for scene understanding and autonomous navigation.

Experiments on publicly available indoor datasets and logistics environments show that the system has a comparative performance on 3D object segmentation while avoiding high computational costs. By employing an efficient object detector and an efficient voxel mapping framework, the system can be extended to a CPU-only robotic platform for real-world application.

A future research direction involves investigating the object-based navigation function, where the mobile robot can receive object information for global navigation and precise docking. Besides, it is valuable to explore camera pose tracking algorithms, which can be integrated into the mapping system to build a semantic map online.

[2] https://github.com/benchun123/object-level-mapping

5 Conclusion

In this chapter, the most important results of this research are summarized. Section 5.1 provides a summary of the thesis, including research questions, methods, results, and lessons learned from the experiments. In Section 5.2, an outlook is presented, and further research areas are identified.

5.1 Summary

The objective of this thesis is to develop a visual localization and mapping system with objects in an intra-logistic environment. The AgiProbot project represents a typical environment specifically designed for the storage, transportation, and management of material handling flows. Within this project, mobile robots are assigned the responsibility of delivering items across different workstations. When equipped with a 2D laser scanner, mobile robots can perceive the surrounding, build a 2D occupancy map, and navigate to designed positions. However, due to the sensor limitation, the reconstructed map contains only 2D information but lacks semantics, which cannot be used for high-level tasks. On the other hand, image-based object detection methods provide rich information about entities present in the scenario, significantly improving capabilities of perception and benefiting intelligent navigation. Therefore, an RGB-D camera is used to detect semantic objects in the environment, estimate camera poses, and reconstruct a dense map with individual objects. The overall research question can be summarized as: **How to detect 3D objects from RGB-D images? Do they benefit localization and mapping?** This research question was divided into three parts:

- **1. How to detect 3D objects from RGB-D images?**
- **2. Does the introduction of semantic objects in the camera tracking process improve localization accuracy?**
- **3. How to efficiently and densely map the environment with semantic objects?**

To answer the first question, three different methods are implemented to detect 3D object detection from a single RGB-D frame, namely the sample-score method, geometry method, and deep learning method. The sample-score method samples object dimension and orientation, generates hundreds of 3D cuboid candidates, and scores them with image features and physical constraints. The best candidate with the highest score can be selected to represent the object. The geometry method converts the RGB-D images into 3D points, removes outliers, and clusters the object points as cuboids. The deep learning method learns the object's attributes from training data and predicts objects with the pre-trained neural network. By designing experiments on the indoor SUN RGB-D dataset and logistics IFL RGB-D dataset, the detection accuracy and runtime efficiency of the three methods are measured: The sample-score method is the fastest, but the deep learning method offers the best accuracy. The geometry method falls in between the other two methods in terms of speed and accuracy.

The second research question was answered by developing a visual SLAM system at the level of objects based on single frame detection results. a point-plane-object SLAM is proposed on top of ORB-SLAM2, where it detects planes and objects in every key-frame, designs a new data association strategy to track them, and integrates them into a unified bundle adjustment framework to jointly optimize camera poses. The proposed method is evaluated on the indoor ICL NUIM dataset and logistics AgiProbot environment. Results show that the point-plane-object SLAM can simultaneously estimate camera poses and build a sparse map with different landmarks. Compared with other state-of-the-art visual SLAM algorithms, the introduction of 3D objects can benefit scene understanding and improve localization accuracy.

For the third research question, an efficient object-level mapping system is proposed, which takes RGB-D sequences as input to build a volumetric object-level map. After solving camera pose tracking and object detection, the detected 3D objects are integrated into a voxel-based mapping framework to incrementally build a global object-level volumetric map. Experiments on the indoor SceneNN dataset and logistics AgiProbot environment show that the proposed system can build a semantic voxel-based map within a robotic platform. When compared to other object-level mapping approaches, the proposed method has a comparative performance on semantic mapping while avoiding high computational costs.

Through experiments, many lessons can be learned:

Lesson 1: Design of Experiments. The design of experiments is a statistical method used to design, plan, and analyze the experiments. It involves controlling the inputs or factors in a process or system, and then measuring the effects of these changes on the output or response variables. A good example is the evaluation of the 3D object detection method on logistics environments (see Section 2.4). In the initial situation. the goal is defined to detect objects as 3D cuboids from a single RGB-D frame. Background, class categories, relative distance, and relative object rotation are four potential factors that may influence the detection results. Then, these parameters are controlled when capturing input for the system, and the response output is measured. By analysing the results, the relative distance and rotation are two majority factors that affect the system. The best object detection results lay within a suitable relative distance range ($0 \sim 3m$) and object rotation ($6 * \pi/8$).

Lesson 2: Sensor Comparison. Different sensors could be used for localization and mapping, such as cameras or laser scanners, each sensor has its own benefits and drawbacks. Cameras are lightweight, low cost, and low power consumption, they can provide rich and robust environmental information and show a good performance on object detection and segmentation. On the other hand, laser scanners offer a precise distance measurement of the surroundings and promise safe navigation for mobile robots. They can be used for both indoor and outdoor environments because of their insensitivity to changes in lighting conditions.

Regarding object-based localization in indoor environments, object-based visual SLAM systems show a good performance on camera localization (see Chapter 3), but they are not as accurate as laser-based systems. However, cameras can significantly improve the perception capabilities to understand the environment. Therefore, in Chapter 4, the laser scanner is chosen to track camera poses but the camera is used to incrementally build a semantic map.

Lesson 3: Real Robotic Platform. Application is valuable for scientific research. When compared to open-source datasets, field experiments are much more challenging because many factors need to be considered, such as hardware setting, sensor configuration, detection limitation, computation resources, etc.

For instance, in Chapter 4, three requirements should be satisfied when seeking the semantic mapping method: the map should contain occupancy information for navigation, the map should provide object information for scene understanding, and the method be efficient for a CPU platform. When considering these constraints, solutions become limited. Ultimately, Voxblox is chosen to efficiently map the environment with occupancy information and extend it to include object information. Furthermore, Other problems are also encountered alongside the mapping algorithm, such as sensor installation, camera calibration, vehicle control, etc. After resolving all the issues, the system can be verified in real-world applications. A comprehensive understanding of the research is obtained, which will lead to a better engineering outcome.

5.2 Outlook

After the research questions are answered and the results are summarized, the boundaries of this thesis are encountered. While exploring visual SLAM with objects, it is recommended that future research focus on the following aspects:

Task-Level Navigation. Since the object-level voxel-based 3D map with RGB-D cameras has been built, it is worth implementing task-level navigation based on the map. On the one hand, the reconstructed TSDF (voxel) map is capable of building Euclidean Signed Distance Fields (ESDF) that can be directly used for path planning in complex environments, such as (Oleynikova et al. 2018) and (Gao et al. 2018). On the other hand, the mapped objects in the map can provide goal information and drive the mobile robot to reach the destination, contributing to a more flexible and intelligent navigation system, such as (Dengler et al. 2021) and (Sivananda et al. 2022).

Deep Learning and Artificial Intelligence With the development of the deep learning method, the neural network tends to be more accurate than geometry methods in object detection, mapping, and other tasks. Limited by the real robotic platform, deep research in this field is not conducted, but it remains an attractive topic for further investigation. For example, Xia et al. (2020) argue that high-level environmental perception can be reached by neural network and benefit semantic localization and mapping. Li et al. (2021a) proposed to solve 3D object detection, association, and mapping tasks in posed RGB videos within a graph neural network, providing a promising direction to combine neural networks and robotics. More related research work can be found in (Chen et al. 2020).

Multi-Sensor System. The current system is implemented on a single RGB-D camera. However, there are still many challenges when applied to real-world applications, such as safety. A multi-sensor system is a good option for industrial products, which can integrate the advantages of different modules. For example, IMU (Inertial Measurement Unit) is also widely used on robots to provide scale and pose constraints for station estimation. More related camera-laser fusion systems can be found in (Bowman et al. 2017).

A Lifting 2D Detection to 3D Cuboid

There are several methods to lift 2D object detection to 3D cuboid, this appendix describes one that is used in the thesis. The fundamental principle is the 2D-3D constraints: the projected vertexes of a 3D cuboid should fit tightly into each side of its 2D detection box.

Given a 2D bounding box, object dimension, and object rotation, the translation of the cuboid can be computed, and a 3D cuboid can be determined. To do that, the perspective camera projection function is formulated as:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{P_{roj}} \begin{pmatrix} \mathbf{X_{3D}} \\ 1 \end{pmatrix}, \tag{A.1}$$

where (u, v) is the pixel position in image, $\mathbf{P_{roj}}$ is camera projection matrix, and $\mathbf{X_{3D}}$ is 3D cuboid vertexes in camera coordinates. Given the cuboid translation $\mathbf{T}$, dimension $\mathbf{D}$ and rotation $\mathbf{R}$, Equation (A.1) can be rewritten as:

$$\begin{aligned} \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \mathbf{P}_{\mathrm{roj}} \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{R}_{3\times3}(\theta) \cdot \mathbf{D}_{3\times1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{T}_{3\times1} \\ 1 \end{pmatrix} \\ &= \mathbf{M}_{3\times4} \begin{pmatrix} \mathbf{T}_{3\times1} \\ 1 \end{pmatrix}, \end{aligned} \tag{A.2}$$

where $\mathbf{M} = \mathbf{P_{roj}} \times (\mathbf{I} \| \mathbf{R} \times \mathbf{D})$. If $\mathbf{M}_i^T = [m_{i0}, m_{i1}, m_{i2}]$, where m_{ij} is the (i, j) elements of the matrix $\mathbf{M}$, the equation can be rewritten as:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{M}_0^T \times \mathbf{T}_{3\times1} + m_{03} \\ \mathbf{M}_1^T \times \mathbf{T}_{3\times1} + m_{13} \\ \mathbf{M}_2^T \times \mathbf{T}_{3\times1} + m_{23} \end{pmatrix}. \tag{A.3}$$

Introducing the 2D bounding box as constraints, for example, the right border of the bounding box x_{max}, the following equation can be obtained:

$$x_{max} = \frac{\mathbf{M}_0^T \times \mathbf{T}_{3\times1} + m_{03}}{\mathbf{M}_2^T \times \mathbf{T}_{3\times1} + m_{23}}, \tag{A.4}$$

$$\left(\mathbf{M}_0^T - \mathbf{M}_2^T \times x_{\max}\right) \times \mathbf{T}_{3\times1} = m_{23} \times x_{\max} - m_{03}. \tag{A.5}$$

It can be seen that one border of the bounding box can formulate one equation to solve for $\mathbf{T}$. Taking four borders, left x_{min}, right x_{max}, top y_{min} and bottom y_{max} into account, then:

$$\begin{pmatrix} \mathbf{M}_0^T - \mathbf{M}_2^T \times x_{\min} \\ \mathbf{M}_1^T - \mathbf{M}_2^T \times y_{\min} \\ \mathbf{M}_0^T - \mathbf{M}_2^T \times x_{\max} \\ \mathbf{M}_1^T - \mathbf{M}_2^T \times y_{\max} \end{pmatrix} \times \mathbf{T}_{3\times1} = \begin{pmatrix} m_{23} \times x_{\min} - m_{03} \\ m_{13} \times y_{\min} - m_{13} \\ m_{23} \times x_{\max} - m_{03} \\ m_{13} \times y_{\max} - m_{13} \end{pmatrix}, \tag{A.6}$$

$$\mathbf{A} \times \mathbf{T}_{3\times1} = \mathbf{b} (\mathbf{b} \neq \mathbf{0}), \tag{A.7}$$

$$\mathbf{T}_{3\times1} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{b}, \tag{A.8}$$

It is an over-constrained system, and the translation $\mathbf{T}$ can be solved by the least-squares method. Each constraint of the 2D bounding box can correspond to any of the 8 corners of the 3D box, which results in $8^4 = 4096$ configurations. As discussed in (Mousavian et al. 2017), when the object is assumed to be parallel to the ground plane, one can narrow the configuration to 64, and choose the minimal error of the least square equation.

B Optimization Problem for SLAM

The meaning of SLAM is to estimate the robot's pose and map the environment at the same time. Theoretically, SLAM can be modelled as a probabilistic problem. Given controls $u_{1:T} = u_1, u_2, ..., u_T$, the robot will move to a new pose, in every pose, the robot can observe landmarks $z_{1:T} = z_1, z_2, ..., z_T$. Since there are some estimation and observation uncertainties, the SLAM system needs to seek the distribution of robot poses $x_{0:T} = x_0, x_1, x_2, ..., x_T$ and map m given the observations and controls, which can be formulated as :

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T}). \tag{B.1}$$

With the help of Bayes' Theory and Kalman Filter, the SLAM problem can be solved. More details can be found Grisetti et al. (2007).

Since it needs to save all distributions of camera poses and landmarks, it is not efficient for a large scene. So, some researchers proposed another graph-based SLAM, which uses a graph to represent the problem. Here, every node corresponds to a pose of the robot x or a location of landmarks m, and every edge between two nodes corresponds to a constraint. Then, the SLAM problem turns to building the graph and finding a node configuration that minimizes the error introduced by the constraints e_{ij}^T, as this term:

$$\mathbf{x}^*, \mathbf{m}^* = \arg\min_{\mathbf{x}} \sum_{i,j} e_{ij}^T \Omega_{ij} e_{ij}^T, \tag{B.2}$$

where the Ω_{ij} represents the observation uncertainty.

B.1 Nonlinear Least-Squares Optimization

Mathematically, the SLAM system needs to find the minimal value of the Equation (B.2), it can be represented as a non-linear least squares problem as follows:

$$x^* = \arg\min_x \frac{1}{2}\|f(x)\|^2, \tag{B.3}$$

$1/2$ is just a coefficient to clarify the following derivation.

Gradient Descent Algorithm $Gradient\ Descent$ algorithm takes every step along the current gradient direction. The Taylor expansion around x for the approximation of $\|f(x)\|^2$ can be expressed as:

$$\|f(x) + \Delta x\|^2 = \|f(x)\|^2 + J(x)\Delta x + \frac{1}{2}\Delta x^\top H \Delta x, \tag{B.4}$$

where J and H represent the first- and second-order derivatives of $\|f(x)\|^2$, they are called $Jacobian\ Matrix$ and $Hessian\ Matrix$, respectively.

$Gradient\ Descent$ Algorithm chooses to remain with the first-order iteration, it is easy to implement, and the convergence speed is fast, but the gradient might be quite small when it is close to minimum. The update rules can be expressed as:

$$\Delta x^* = -J^\top(x), \tag{B.5}$$

Newton's Algorithm Another approach is $Newton's\ Algorithm$, which chooses to stay with the second order iteration of Equation (B.4). At each iteration, it takes a step towards the minimum of the quadratic function, which is much faster than $Gradient\ Descent$ Algorithm. However, for general high-dimensional cost functions, it is hard to compute the H. The update rule is

$$H\Delta x^* = -J^\top(x), \tag{B.6}$$

Gauss-Newton Algorithm $Gauss-Newton$ Algorithm is the third approach that can solve the nonlinear optimization problem efficiently. Instead of approximating $\|f(x)\|^2$, $f(x)$ can be approximated by a first-order Taylor expansion around x:

$$f(x + \Delta x) \approx f(x) + J(x)\Delta x, \tag{B.7}$$

here, where $J(x)$ represents the first-order derivatives of $f(x)$ not $\|f(x)\|^2$.

So, the original non-linear least squares problem can be approximated as:

$$\Delta x^* = \arg\min_{\Delta x} \frac{1}{2}\|f(x) + J(x)\Delta x\|^2, \tag{B.8}$$

which seeks a deviation Δx that minimizes an approximation to the cost in a neighbourhood of x. Then, Equation (B.9) can be expanded as:

$$\begin{aligned}\frac{1}{2}\|f(x) + J(x)\Delta x\|^2 &= \frac{1}{2}(f(x) + J(x)\Delta x)^T(f(x) + J(x)\Delta x) \\ &= \frac{1}{2}\left(\|f(x)\|^2 + 2f(x)^T J(x)\Delta x + \Delta x^T J(x)^T J(x)\Delta x\right)\end{aligned} \tag{B.9}$$

The optimal update step Δx can be computed by finding the minimum of the above function:

$$\begin{aligned}2J(x)^T f(x) + 2J(x)^T J(x)\Delta x &= 0 \\ J(x)^T J(x)\Delta x &= -J(x)^T f(x)\end{aligned} \tag{B.10}$$

To summarize, the process and update rule for $Gauss - Newton$ Algorithm is:

$$\begin{aligned}\Delta x &= -\left(J(x)^T J(x)\right)^{-1} J(x)^T f(x) \\ x &\leftarrow x + \Delta x\end{aligned} \tag{B.11}$$

Levenberg-Marquardt (L-M) Algorithm $Gauss - Newton$ Algorithm performs well when it is close to the minimum, but the cost may not decrease at each iteration, instead, $Gradient\ Descent$ has the advantages of convergence. A smart idea is to combine them together, which is called $L - MAlgorithm.$ This algorithm introduces a trust region to switch these two algorithms. With this idea, Equation (B.10) can be modified as:

$$(J(x)^T J(x) + \lambda \mathbf{I})\Delta x = -J(x)^\top f(x) \tag{B.12}$$

where λ is a damping ratio and $\mathbf{I}$ is an identity matrix. When $\lambda \to \infty$, $\Delta \mathbf{x} \approx -J(x)^\top f(x)$, it changes to $Gradient\ Descent\ Algorithm.$ When $\lambda \to 0$, indicating to follow the standard $Gauss - Newton\ Algorithm.$

B.2 Non-Euclidean Optimization

As shown in Equation (B.11), the last step for iterative optimization is the variable update. However, $\mathbf{x} \leftarrow \mathbf{x} + \Delta \mathbf{x}$ only applies to Euclidean space and is not suitable for non-Euclidean space, such as 3D rotation. In SLAM optimization, the 3D rotation is represented as Euler angles or rotation vectors, it doesn't follow the update rules, instead, the update is performed as $\mathbf{R}(\mathbf{x}) \cdot \mathbf{R}(\Delta \mathbf{x})$. In this case, Lie Algebra can be introduced to compute non-Euclidean optimization. For more details, please refer to Blanco-Claraco (2021).

List of Figures

List of Tables

List of Publications

Journal articles

Benchun Zhou, Maximilian Gilles, and Yongqi Meng. Structure slam with points, planes and objects. *Advanced Robotics*, 36(20):1060–1075, 2022.

Benchun Zhou, Kai Furmans, Fu Yunxiang, and Pang Hao. Efficient object-level semantic mapping with rgb-d cameras. *Autonomous Robots(Under Review, Submitted at 23 Jun 2023)*, 2024.

Conference contributions

Benchun Zhou, Aibo Wang, Jan-Felix Klein, and Furmans Kai. Object detection and mapping with bounding box constraints. In *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 1–6. IEEE, 2021.

Benchun Zhou, Yongqi Meng, and Furmans Kai. Object-based loop closure with directional histogram descriptor. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1346–1351. IEEE, 2022.

Benchun Zhou, Jan-Felix Klein, Bo Wang, and Markus Hillemann. Semantic mapping and autonomous navigation for agile production system. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023.

Bibliography

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). *Surf: Speeded up robust features*, volume 3951, pages 404–417. Springer Berlin Heidelberg.

Blanco-Claraco, J. L. (2021). A tutorial on $\mathbf{SE}(3)$ transformation parameterizations and on-manifold optimization. *ArXiv preprint ArXiv:2103.15980*.

Bowman, S. L., Atanasov, N., Daniilidis, K., and Pappas, G. J. (2017). Probabilistic data association for semantic slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729. IEEE.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). *Brief: Binary robust independent elementary features*, pages 778–792. Springer Berlin Heidelberg.

Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890.

Censi, A. (2008). An icp variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*, pages 19–25. IEEE.

Chen, C., Wang, B., Lu, C. X., Trigoni, N., and Markham, A. (2020). A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *ArXiv preprint ArXiv:2006.12567*.

Chen, S., Song, S., Zhao, J., Feng, T., Ye, C., Xiong, L., and Li, D. (2021). Robust dual quadric initialization for forward-translating camera movements. *IEEE Robotics and Automation Letters*, 6(3):4712–4719.

Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 941–947. IEEE.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 303–312.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

Dengler, N., Zaenker, T., Verdoja, F., and Bennewitz, M. (2021). Online object-oriented semantic mapping and map updating. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE.

Dwibedi, D., Malisiewicz, T., Badrinarayanan, V., and Rabinovich, A. (2016). Deep cuboid detection: Beyond 2d bounding boxes. *ArXiv preprint ArXiv:1611.10010.*

Furmans, K., Seibold, Z., and Trenkle, A. (2018). *Future technologies in intralogistics and material handling*, page 545–574. Springer Cham.

Gao, F., Wu, W., Lin, Y., and Shen, S. (2018). Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351. IEEE.

Garon, M., Boulet, P.-O., Doiron, J.-P., Beaulieu, L., and Lalonde, J.-F. (2016). Real-time high resolution 3d data on the hololens. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 189–191. IEEE.

Gawel, A., Del Don, C., Siegwart, R., Nieto, J., and Cadena, C. (2018). X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3):1687–1694.

Grinvald, M., Furrer, F., Novkovic, T., Chung, J. J., Cadena, C., Siegwart, R., and Nieto, J. (2019). Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044.

Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

Guo, X., Hu, J., Chen, J., Deng, F., and Lam, T. L. (2021). Semantic histogram based graph matching for real-time multi-robot global localization in large scale environment. *IEEE Robotics and Automation Letters*, 6(4):8349–8356.

Gupta, A., Hebert, M., Kanade, T., and Blei, D. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *Advances in Neural Information Processing Systems*, 23.

Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531. IEEE.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969.

Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1849–1856. IEEE.

Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE.

Himstedt, M. and Maehle, E. (2017). Online semantic mapping of logistic environments using rgb-d cameras. *International Journal of Advanced Robotic Systems*, 14(4):1729881417720781.

Holz, D. and Behnke, S. (2013). *Fast range image segmentation and smoothing using approximate surface reconstruction and region growing*, volume 12, pages 61–73. Springer Berlin Heidelberg.

Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2011). *Real-time plane segmentation using RGB-D cameras*, pages 306–317. Springer Berlin Heidelberg.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34:189–206.

Hosseinzadeh, M., Latif, Y., Pham, T., Suenderhauf, N., and Reid, I. (2019). *Structure aware SLAM using quadrics and planes*, pages 410–426. Springer Cham.

Hosseinzadeh, M., Latif, Y., and Reid, I. (2017). Sparse point-plane slam. In *Australasian Conference on Robotics and Automation*.

Hsiao, M., Westman, E., Zhang, G., and Kaess, M. (2017). Keyframe-based dense planar slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5110–5117. IEEE.

Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., and Yeung, S.-K. (2016). Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE.

Huang, S., Chen, Y., Yuan, T., Qi, S., Zhu, Y., and Zhu, S.-C. (2019). Perspectivenet: 3d object detection from a single rgb image via perspective points. *Advances in Neural Information Processing Systems*, 32.

Huang, S., Qi, S., Xiao, Y., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2018). Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. *Advances in Neural Information Processing Systems*, 31.

Jiang, H. and Xiao, J. (2013). A linear approach to matching cuboids in rgbd images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2171–2178.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868.

Kaess, M. (2015). Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611. IEEE.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE.

Klein, J.-F., Wurster, M., Stricker, N., Lanza, G., and Furmans, K. (2021). Towards ontology-based autonomous intralogistics for agile remanufacturing production systems. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 01–07. IEEE.

Kohlbrecher, S., Von Stryk, O., Meyer, J., and Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160. IEEE.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g 2 o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE.

Lategahn, H., Geiger, A., and Kitt, B. (2011). Visual slam for autonomous ground vehicles. In *2011 IEEE International Conference on Robotics and Automation*, pages 1732–1737. IEEE.

Le, P.-H. and Košecka, J. (2017). Dense piecewise planar rgb-d slam for indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4944–4949. IEEE.

Leigh, A., Pineau, J., Olmedo, N., and Zhang, H. (2015). Person tracking and following with 2d laser scanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 726–733. IEEE.

Lemaire, T., Berger, C., Jung, I.-K., and Lacroix, S. (2007). Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364.

Li, A., Ruan, X., Huang, J., Zhu, X., and Wang, F. (2019a). Review of vision-based simultaneous localization and mapping. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 117–123. IEEE.

Li, J., Koreitem, K., Meger, D., and Dudek, G. (2020a). View-invariant loop closure with oriented semantic landmarks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7943–7949. IEEE.

Li, J., Meger, D., and Dudek, G. (2019b). Semantic mapping for view-invariant relocalization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7108–7115. IEEE.

Li, K., DeTone, D., Chen, Y. F. S., Vo, M., Reid, I., Rezatofighi, H., Sweeney, C., Straub, J., and Newcombe, R. (2021a). Odam: object detection, association, and mapping using posed rgb video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5998–6008.

Li, W., Gu, J., Chen, B., and Han, J. (2020b). Incremental instance-oriented 3d semantic mapping via rgb-d cameras for unknown indoor scene. *Discrete Dynamics in Nature and Society*, 2020.

Li, Y., Yunus, R., Brasch, N., Navab, N., and Tombari, F. (2021b). Rgb-d slam with structural regularities. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11581–11587. IEEE.

Liao, Z., Hu, Y., Zhang, J., Qi, X., Zhang, X., and Wang, W. (2022). So-slam: Semantic object slam with scale proportional and symmetrical texture constraints. *IEEE Robotics and Automation Letters*, 7(2):4008–4015.

Liao, Z., Wang, W., Qi, X., and Zhang, X. (2020a). Rgb-d object slam using quadrics for indoor environments. *Sensors*, 20(18):5150.

Liao, Z., Wang, W., Qi, X., Zhang, X., Xue, L., Jiao, J., and Wei, R. (2020b). Object-oriented slam using quadrics and symmetry properties for indoor environments. *ArXiv preprint ArXiv:2004.05303*.

Lin, S., Wang, J., Xu, M., Zhao, H., and Chen, Z. (2021). Topology aware object-level semantic mapping towards more robust loop closure. *IEEE Robotics and Automation Letters*, 6(4):7041–7048.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

Liu, C., Kim, K., Gu, J., Furukawa, Y., and Kautz, J. (2019a). Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459.

Liu, C., Yang, J., Ceylan, D., Yumer, E., and Furukawa, Y. (2018). Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588.

Liu, K., Fan, Z., Liu, M., and Zhang, S. (2019b). Object-aware semantic mapping of indoor scenes using octomap. In *2019 Chinese Control Conference (CCC)*, pages 8671–8676. IEEE.

Liu, Y., Petillot, Y., Lane, D., and Wang, S. (2019c). Global localization with object-level semantics and topology. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4909–4915. IEEE.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

Mascaro, R., Teixeira, L., and Chli, M. (2022). Volumetric instance-level semantic mapping via multi-view 2d-to-3d label diffusion. *IEEE Robotics and Automation Letters*, 7(2):3531–3538.

Mayershofer, C., Holm, D.-M., Molter, B., and Fottner, J. (2020). Loco: Logistics objects in context. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 612–617. IEEE.

McCormac, J., Handa, A., Davison, A., and Leutenegger, S. (2017). Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE.

Meng, Y. and Zhou, B. (2022). Ellipsoid slam with novel object initialization. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1333–1338. IEEE.

Mishima, M., Uchiyama, H., Thomas, D., Taniguchi, R.-i., Roberto, R., Lima, J. P., and Teichrieb, V. (2019). Incremental 3d cuboid modeling with drift compensation. *Sensors*, 19(1):178.

Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE.

Nicholson, L., Milford, M., and Sünderhauf, N. (2018). Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8.

Nie, Y., Han, X., Guo, S., Zheng, Y., Chang, J., and Zhang, J. J. (2020). Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 55–64.

Ok, K., Liu, K., Frey, K., How, J. P., and Roy, N. (2019). Robust object-based slam for high-speed autonomous navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 669–675. IEEE.

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., and Nieto, J. (2017). Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. IEEE.

Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J. (2018). Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles. *IEEE Robotics and Automation Letters*, 3(3):1474–1481.

Oliva, A. and Torralba, A. (2006). Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 155:23–36.

Pang, L., Cao, Z., Yu, J., Liang, S., Chen, X., and Zhang, W. (2019). An efficient 3d pedestrian detector with calibrated rgb camera and 3d lidar. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2902–2907. IEEE.

Parkhiya, P., Khawad, R., Murthy, J. K., Bhowmick, B., and Krishna, K. M. (2018). Constructing category-specific models for monocular object-slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4517–4524. IEEE.

Pham, Q.-H., Hua, B.-S., Nguyen, T., and Yeung, S.-K. (2019). Real-time progressive 3d semantic segmentation for indoor scenes. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1089–1098. IEEE.

Pham, T. T., Reid, I., Latif, Y., and Gould, S. (2015). Hierarchical higher-order regression forest fields: An application to 3d indoor scene labelling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2246–2254.

Qi, C. R., Chen, X., Litany, O., and Guibas, L. J. (2020). Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4404–4413.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271. IEEE.

Rosinol, A., Abate, M., Chang, Y., and Carlone, L. (2020). Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE.

Rosten, E. and Drummond, T. (2006). *Machine learning for high-speed corner detection*, pages 430–443. Springer Berlin Heidelberg.

Rubino, C., Crocco, M., and Del Bue, A. (2017). 3d object localisation from multi-view image detections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1281–1294.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359.

Sivananda, K. P., Verdoja, F., and Kyrki, V. (2022). Augmented environment representations with complete object models. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1123–1130. IEEE.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576.

Stumm, E., Mei, C., Lacroix, S., Nieto, J., Hutter, M., and Siegwart, R. (2016). Robust visual place recognition with graph kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4535–4544.

Sünderhauf, N., Pham, T. T., Latif, Y., Milford, M., and Reid, I. (2017). Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085. IEEE.

Taguchi, Y., Jian, Y.-D., Ramalingam, S., and Feng, C. (2013). Point-plane slam for hand-held 3d sensors. In *2013 IEEE International Conference on Robotics and Automation*, pages 5182–5189. IEEE.

Tian, R., Zhang, Y., Feng, Y., Yang, L., Cao, Z., Coleman, S., and Kerr, D. (2021). Accurate and robust object-oriented slam with 3d quadric landmark construction in outdoor environment. *ArXiv Preprint ArXiv:2110.08977*.

Trevor, A. J., Gedikli, S., Rusu, R. B., and Christensen, H. I. (2013). Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, pages pp–1.

Wang, J., Wang, P., and Chen, Z. (2018). A novel qualitative motion model based probabilistic indoor global localization method. *Information Sciences*, 429:284–295.

Wang, J., Wang, P., Dai, D., Xu, M., and Chen, Z. (2020). Regression forest based rgb-d visual relocalization using coarse-to-fine strategy. *IEEE Robotics and Automation Letters*, 5(3):4431–4438.

Xia, L., Cui, J., Shen, R., Xu, X., Gao, Y., and Li, X. (2020). A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, 17(3):1729881420919185.

Xiao, J., Russell, B., and Torralba, A. (2012). Localizing 3d cuboids in single-view images. *Advances in Neural Information Processing Systems*, 25.

Xie, Y., Rambach, J., Shu, F., and Stricker, D. (2021). Planesegnet: Fast and robust plane estimation using a single-stage instance segmentation cnn. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13574–13580. IEEE.

Yang, F. and Zhou, Z. (2018). Recovering 3d planes from a single image via convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100.

Yang, S. and Scherer, S. (2019a). Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938.

Yang, S. and Scherer, S. (2019b). Monocular object and plane slam in structured environments. *IEEE Robotics and Automation Letters*, 4(4):3145–3152.

Yang, S., Song, Y., Kaess, M., and Scherer, S. (2016). Pop-up slam: Semantic monocular plane slam for low-texture environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1222–1229. IEEE.

Zaenker, T., Verdoja, F., and Kyrki, V. (2020). Hypermap mapping framework and its application to autonomous semantic exploration. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 133–139. IEEE.

Zhang, Q. and Pless, R. (2004). Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2301–2306. IEEE.

Zhang, X., Wang, W., Qi, X., Liao, Z., and Wei, R. (2019). Point-plane slam using supposed planes for indoor environments. *Sensors*, 19(17):3795.

Zinßer, T., Schmidt, J., and Niemann, H. (2005). Point set registration with integrated scale estimation. In *International Conference on Pattern Recognition and image processing*, pages 116–119.