Master's Thesis

A Real-Time Semantic-Aware Simultaneous Localization and Mapping for Unmanned Aerial Vehicles

NGUYEN CANH Thanh

Supervisor: Professor CHONG Nak Young

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

August 2024

Abstract

Unmanned Aerial Vehicles (UAVs) are essential in various fields such as disaster monitoring, environmental surveys, search and rescue missions, and infrastructure inspection, where a precise understanding of the surrounding environments is crucial. A fundamental challenge for UAVs operating in dynamic and unstructured environments is the capability to navigate and map their surroundings in real-time accurately. Simultaneous Localization and Mapping (SLAM) is an essential technology that addresses this challenge by enabling UAVs to construct detailed maps while simultaneously determining their location within these maps. Visual SLAM has become increasingly crucial in the effective localization and representation of a map consisting of 3D points; however, it lacks semantic information and serves for high-level tasks. Numerous previous approaches have aimed to build dense maps; however, these reconstructions are still just aggregates of points, and thus lack any supplementary semantic information or relationships. A robot, moreover, must be capable of mapping its environment, localizing itself within that map, and comprehending the semantic information of the surrounding scene. Semantic SLAM addresses three fundamental tasks simultaneously, aiming to produce the most precise and comprehensive environmental model in an environment. Achieving this requires a careful balance between the accuracy of the semantic map and the memory resources it consumes. Over recent decades, semantic SLAM has garnered increasing interest and has been explored in various ways by different research communities, driven by the goal of practical deployment in a real-world application. The broad interest has expanded the problem's scope and provided diverse perspectives, leading to numerous approaches based on various theories and concepts. However, this has also created a disconnect between research paths that could be mutually beneficial. However, integrating localization, semantic segmentation, and 3D reconstruction simultaneously poses a significant challenge, particularly for UAVs that operate with limited power and computational capacities.

To mitigate these challenges, this thesis emphasizes the development of Semantic SLAM systems that integrate metric environment structures with semantic object information to create comprehensive semantic maps. Our proposed method is organized into two key enhancements: a 3D semantic mapping method and a 2.5D probabilistic metric map approach. By integrating semantic details, we aim to enhance the effectiveness of the semantic SLAM system. Initially, we introduce an innovative strategy

to tackle the issues related to the extraction and use of semantic data in UAV operations. Our framework combines cutting-edge visual SLAM for accurate 6-DoF pose estimation with sophisticated object segmentation techniques at the back end. To enhance the framework's computational and storage efficiency, we employ a simplified voxel-based 3D map representation known as OctoMap for system construction. Additionally, we integrate a fusion algorithm to retrieve semantic information from each frame in the front-end SLAM task and the associated point. Secondly, we propose to construct a probabilistic metric map enriched with object data from RGB-D images. This method integrates a cutting-edge YOLOv8-based object detection framework upfront and a 2D SLAM method - Cartographer, at the back end. To track and position semantic object categories obtained from the front-end interface, we utilize the innovative BoT-SORT methodology. A new association technique is proposed to determine objects' positions and project them onto the metric map. Unlike previous studies, our method focuses on navigating environments that contain various hollow objects on the bottom. The output is a probabilistic map that significantly enhances the representation of the map by incorporating object-specific details, including class distinctions, accurate positions, and object heights.

To demonstrate the pose estimation performance of our semantic SLAM system, we performed evaluations using two different types of datasets: 1) publicly available TUM real-world RGB-D data sequences and 2) a Gazebo simulation dataset. We assessed the precision of 6-DoF pose estimation using the Root Mean Square Error (RMSE) of Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). Compared to the state-of-the-art visual SLAM - ORB-SLAM2, the results demonstrated precise pose estimation and smooth movement within environments. Our mapping outcomes show that our system greatly enhances mapping accuracy, computational efficiency, and the UAVs' capability to autonomously navigate complex scenarios. Additionally, our system is tested on an embedded computer - Jetson Xavier AGX unit, to illustrate its effectiveness in real-world applications.

In summary, this thesis presents the development of advanced Semantic SLAM systems designed to enhance the autonomous capabilities of UAVs in dynamic and unstructured environments. By integrating semantic understanding and probabilistic mapping methodologies, our approach significantly improves mapping accuracy, computational efficiency, and UAV navigation. Future work could focus on building a sustainable 3D active semantic SLAM system that provides essential information for UAV applications, ensuring the safe operation of flying equipment, and enhancing localization capabilities. This approach takes advantage of the full potential of camera data, making additional sensors such as GPS or LiDAR unnecessary.

Keywords: Simultaneous Localization and Mapping (SLAM), Semantic SLAM, Semantic Mapping, Unmanned Aerial Vehicles (UAVs), Robotics.

Acknowledgments

I am profoundly grateful to everyone who has provided support during my master's studies. Pursuing my Master's degree at Information Science School of JAIST has been a life-changing experience, one which was made achievable through the support and direction of these individuals.

First and foremost, I am profoundly grateful to my supervisor, Professor **Chong Nak Young** of the Japan Advanced Institute of Science and Technology (JAIST) for his invaluable guidance, unwavering support, and insightful feedback during my Master course. After arriving at JAIST in October 2023 and being accepted as a Master's student in the Robotics Lab by Prof. Chong, I benefited greatly from his extensive supervision. Expressing some experiences is difficult, but only someone inside could appreciate his constant presence, listening, patience, motivation, and boundless enthusiasm and expertise. His constructive feedback and financial support helped me overcome significant research obstacles and inspired me to explore new ideas. Being inspired by his work styles and progressively gaining valuable research experience from him enables me to approach academic tasks with confidence. Once again, I struggle to convey just *"how critical his roles have been to my academic journey!"*. Therefore, I am incredibly grateful to Prof. **Chong Nak Young** for his enthusiastic support.

I would like to extend my heartfelt thanks to Professor **Xiem HoangVan** from VNU - University of Engineering and Technology for his continued support and for providing a stimulating academic environment. Prof. Xiem was my initial supervisor in Vietnam, and his comments have elevated me from a novice kid to a more knowledgeable student in the field. In addition, I would also like to thank my second supervisor Professor **Masato Suzuki**, my minor research project Professor **Kiyofumi Tanaka** for his valuable feedback. Furthermore, I am extremely thankful to Professor **Nguyen Le Minh** and Professor **Mizuhito Ogawa**, whose guidance and support provided me the opportunity to study at JAIST.

I wish to express my sincere gratitude to my mentor, Mr. Zhou Peiwei, as well as my colleagues in Chong's laboratory for their extensive support in various aspects of my professional and personal life. My appreciation also goes to my friends, especially the members of the Vietnamese Association in JAIST, who have continually inspired and motivated me.

I am immensely grateful to my mother for her love, patience, and understanding. Her unwavering support and unconditional love have been

my anchor throughout this journey. I love you from the bottom of my heart.

Thank you all for your contributions and encouragement. This achievement would not have been possible without you.

List of Abbreviations and Symbols

List of Symbols

(ϕ, θ, ψ)	Robot orientation
$\mathbf{D}_t$	Data association at time t
$\mathbf{L}_t$	Landmark state at time t
$\mathbf{p} = (x, y, z)$	Robot position
$\mathbf{p}_c^2, \mathbf{p}_c^3$	2D feature points and corresponding 3D point
$\mathbf{R}$	Rotation matrix
$\mathbf{T}$	Transformation matrix
$\mathbf{t}$	Translation matrix
$\mathbf{u}$	Control input
$\mathbf{W}$	Set of waypoints
$\mathbf{w}_t$	Noise at time t
$\mathbf{X}_t$	Robot state at time t
$\mathbf{Z}_t$	Observation at time t
$\mathcal{D}$	Set of all data associations
$\mathcal{L}$	Set of all landmarks state
$\mathcal{X}$	Set of all robot states
$\mathcal{Z}$	Set of all observations
${}^o\mathbf{R}_c, {}^o\mathbf{T}_c, {}^o\mathbf{M}_c$	The rotation matrix, the translation matrix, and the transformation matrix from world coordinate frame O to camera coordinate frame C, respectively

$S0(3)$	Three-dimensional Special Orthogonal group
$SE(3)$	Three-dimensional Special Euclidean group

List of Abbreviations

ATE	Absolute Trajectory Error
CNN	Convolutional Neural Network
FOV	Field of View
GNSS	Global Navigation Satellite System
IMU	Inertial measurement unit
LiDAR	Light Detection and Ranging
PSP Net	Pyramid Scene Parsing Network
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
ROS	Robot Operating System
RPE	Relative Pose Error
SLAM	Simualteneous Localization and Mapping
UAV	Unmanned Aerial Vehicle

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this chapter, we introduce the foundational concepts, the overview, motivation, objective, and approach of our problem addressed in this thesis. This chapter concludes with a summary of our contributions and the structure of this thesis.

1.1 Overview

In this rapidly evolving field of robotics and autonomous systems, Drones, or Unmanned Aerial Vehicles (UAVs), have played a growing role in gathering geoinformation for applications such as firefighting rescue, inspections, and agriculture. Due to the ability to navigate through challenging indoor or outdoor environments, UAVs can be effective in identifying objects and people on the ground within the rubble, as well as broadcasting evacuation messages to facilitate direct communication with victims with the support of various sensors [1]. The ability of UAVs to accurately perceive and navigate their surroundings in real-time is essential for these applications. SLAM (Simultaneous Localization and Mapping) is a critical technology that allows UAVs to construct detailed maps while concurrently determining their location within these maps. The SLAM system encompasses a front-end component that processes and utilizes input sensors, along with a back-end component dedicated to optimization, as illustrated in Figure 1.1. Based on various kinds of sensors such as Camera, lidar, gnss and imu, SLAM can split into two main approaches: L-SLAM and vSLAM. L-SLAM utilized lidar sensors as an input, which is attracting attention from researchers due to the effective and various open source [2]. However, there are still challenges, especially in optimizing processing speed and losing track of the robot in places with few obstacles, because it is difficult to align cloud points in such areas. On the other hand, vSLAM reconstructs the surrounding environment map by camera, which is cheaper and also provides more useful information than lidar. In the vSLAM system, the localization component focuses on identifying the camera's pose of the camera or its trajectory, known as Visual

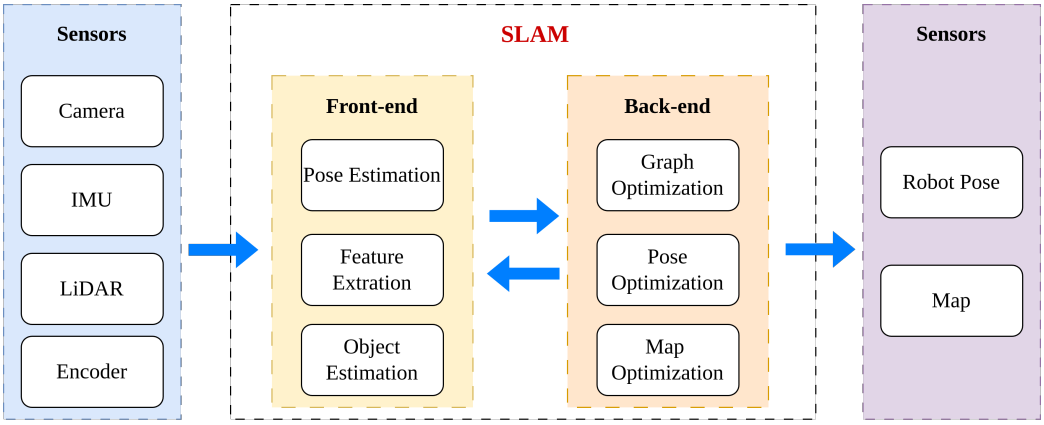


Figure 1.1: Overview of SLAM Components

Odometry, without the need to maintain or recover a set of rigid landmarks on the map [3]. The relative poses of successive camera frames can be estimated by marginalizing out the landmarks. However, in geometric reconstruction, the focus shifts to estimating the positions of the camera and the landmarks within the map [4]. The objective of the vSLAM challenge is to incrementally generate 2D or 3D maps and determine the camera's poses from images in real-time, rendering them suitable for applications involving robotic vision and real-time processing.

This thesis emphasizes the reconstruction problem within the scope of vSLAM. Consider the vSLAM problem with robot state $\mathcal{X} \triangleq \{\mathbf{X}_t\}_{t=1}^{T} \in SE(3)$ at time t and discrete-time deterministic kinematics. In general, the problem of data association in vSLAM is to determine the most likely collection of robot poses $\mathcal{X}$ and landmark states $\mathcal{L} \triangleq \{L_m\}_{m=1}^{M}$ of M static landmarks given observation measurements made in the robot state $\mathcal{Z} \triangleq \{\mathbf{Z}_t\}_{t=1}^{T}$, as the robot navigates through an unfamiliar environment. Therefore, the solution to the SLAM problem involves estimating landmarks and robot poses using a maximum likelihood method:

$$\mathcal{X}_{ml}^{*}, \mathcal{L}_{ml}^{*} = \underset{\mathcal{X},\mathcal{L}}{\operatorname{argmax}}\, p(\mathcal{Z}|\mathcal{X}, \mathcal{L}, \mathcal{S}) \tag{1.1}$$

Define $\mathcal{D} \triangleq \{\mathbf{D}_t\}_{t=1}^{T}$ as the set representing the association of all measurements $\mathbf{Z}_t$. One widely-used method to address (1.1) is the expectation-maximization algorithm, which iterates between the probability of data

association and the log-likelihood conditioned on the data:

$$\begin{aligned}\mathcal{X}^{i+1}, \mathcal{L}^{i+1} &= \underset{\mathcal{X},\mathcal{L},\mathcal{D}}{\operatorname{argmax}} \mathbb{E}_{\mathcal{D}}[\log p(\mathcal{Z}|\mathcal{X}, \mathcal{L}, \mathcal{D})|\mathcal{X}^i, \mathcal{L}^i, \mathcal{Z}] \\ &= \underset{\mathcal{X},\mathcal{L},\mathcal{D}}{\operatorname{argmax}} \sum_{\mathcal{D}\in\mathbb{D}} p(\mathcal{D}|\mathcal{X}^i, \mathcal{L}^i, \mathcal{Z}) \log p(\mathcal{Z}|\mathcal{X}, \mathcal{L}, \mathcal{D})\end{aligned} \tag{1.2}$$

vSLAM has proven effective in localization and the representation of environments as 3D point clouds. In the context of vSLAM, maps can generally be classified into two types: 2D maps and 3D maps. Both types play critical roles in robotic navigation and environmental understanding, but they differ in their complexity, computational requirements, and the level of detail they provide. 2D maps are simpler representations that capture the layout of an environment in two dimensions. These maps are often used for basic navigation tasks and are sufficient for many applications where detailed depth information is not necessary. The advantages of 2D maps include: (1) Lower Computational Requirements: Generating and maintaining 2D maps require less computational power and memory, making them suitable for UAVs and robots with limited resources; (2) Ease of Use: 2D maps are easier to visualize and interpret, which can be beneficial for human operators who need to understand the robot's environment quickly; (3) Fast Processing: The algorithms for 2D SLAM are generally faster, allowing for real-time performance in various applications. However, 2D maps have significant limitations, particularly in complex environments where height and depth information are crucial. For example, in indoor environments with multiple floors, furniture, and other obstacles, a 2D map cannot accurately represent the spatial relationships; heights of different objects, and hollow bottom objects. This can lead to navigation errors and inefficient path planning. 3D maps provide a more detailed depiction of the environment by capturing the structure and spatial relationships in three dimensions. These maps are particularly useful for various use cases that need comprehensive environmental understanding, like obstacle avoidance and advanced navigation. The benefits of 3D maps include: (1) Detailed Spatial Representation: 3D maps can capture the height, width, and depth of objects, providing a more comprehensive and precise depiction of the environment; (2) Improved navigation: With 3D information, UAVs and robots can navigate more effectively around obstacles, through narrow passages, and over varied terrain; (3) Enhanced Scene Understanding: 3D maps enable better recognition and understanding of the environment, facilitating tasks like object detection, scene reconstruction, and interaction with objects. However, 3D maps also have their drawbacks. They require more computational resources to generate and maintain, which can be challenging

for UAVs and robots with limited processing power. Furthermore, processing and visualizing 3D data can be more complex, requiring advanced algorithms and techniques. In this thesis, we focus on the 3D map and enhance the 2D map by incorporating semantic information to strike a balance between the simplicity of 2D maps and the richness of 3D maps.

However, traditional visual SLAM lacks semantic information, which is necessary for higher-level tasks and comprehensive environmental understanding. To address this limitation, incorporating semantic information into the SLAM system has become a significant research focus. Semantic SLAM enhances traditional SLAM by integrating object recognition and scene understanding, enabling UAVs to execute more sophisticated and intelligent operations. Conversely, the development of artificial intelligence, particularly Convolutional Neural Networks (CNNs), has revolutionized semantic information extractions from visual data. CNNs can accurately identify and classify objects within a scene, providing valuable semantic information that can be integrated with geometric data from SLAM systems. By leveraging CNNs, we can create a more informative and useful map representation, combining both geometric and semantic data

This thesis focuses on creating a resilient Semantic SLAM system that integrates metric mapping with semantic data of objects. Specifically, we focus on sparse real-time SLAM methods due to their computational efficiency and scalability for large-scale applications. Our proposed system utilizes a localized RGB-D camera to incrementally construct an object-level map, employing Visual Odometry (VO) for pose estimation and object instance segmentation for creating semantic maps. Additionally, our system aims to address the challenge of hollow bottom objects in indoor environments. Traditional SLAM systems often represent objects as points or paths, resulting in inaccurate navigation decisions. We propose a 2.5D probabilistic semantic map that fuses metric environment structure with height information, enabling UAVs to understand and interact with their environment more intelligently.

We implement our Semantic SLAM system using the Robot Operating System (ROS) [5] framework, ensuring modularity and ease of integration with existing robotics platforms. ROS provides a robust and flexible infrastructure for developing and deploying SLAM systems, facilitating real-time processing and data exchange between various components.

In summary, our thesis aims to advance the field of UAV navigation and mapping by developing an efficient and accurate Semantic SLAM system. By integrating semantic information with geometric mapping, we enhance the UAV's ability to navigate complex environments and perform high-level tasks autonomously. This research has significant implications for

various applications, including search and rescue missions, inspections, and environmental monitoring.

1.2 Motivation and Objectives

Geometric understanding or environmental reconstruction involves inferring and retrieving the scene's structure from image frames with respect to the modeled scene map during localization. Conversely, understanding the semantic scene entails extracting higher-level semantics or relational information. For effective reconstruction, the model required sufficient geometric and photometric correspondences across multiple views to recover lost information. In addition, the density of these correspondences leads to varied map representations, from sparse to semi-dense and dense maps. Although dense map reconstruction encompasses larger portions of the environment, it is highly dependent on the quality of per-pixel matching. Despite recent advancements in deep learning, limited texture and poor photometric conditions or image distortions constrain its application to small room-scale applications. Moreover, these methods have high computational and memory requirements restricting them to mostly non-real-time scenarios. In contrast, sparse representations are more computationally manageable, robust, and scaleable to large-scale applications, making them prevalent in autonomous robots and augmented reality applications. This thesis specifically focuses on the sparse real-time SLAM methods due to their advancements.

Semantic scene understanding comprises various techniques used to extract high-level information from the captured scene, such as indoor furniture, outdoor structures, human presence, action recognition, etc. These various applications can be broadly categorized into three principal groups: object detection, image classification, and semantic segmentation. The intersection of scene understanding with geometric localization has been a focal point of research in both computer vision and robotics. For example, in a rescue operation, a robot depending purely on a conventional SLAM-created map faces difficulties in executing intricate tasks like: "navigating around the desk to find a victim near the bed". Semantic mapping, which integrates environmental geometry estimation with semantic annotation, surpasses traditional geometric mapping, improving UAVs' situational awareness and engagement. This approach employs traditional vSLAM as a foundation for map constructing and localizing it in the surrounding environment and subsequently executes semantic scene understanding in post-processing to interpret the high-level details of the map. Overall, this task remains challenging due to: (1) the unreliability of indoor GPS, (2) the cluttered

nature of environments, (3) the necessity for real-time processing, and (4) the complexity of semantic maps. Thus, our research aims to create semantic data mapping, which is essential for UAVs to carry out sophisticated autonomous functions. In this thesis, we focus on an efficient Semantic Segmentation Sparse Mapping SLAM system to iteratively build an object-level map using a localized RGB-D camera. The proposed system consists of two main components: an RGB-D SLAM framework based on propagation using VO estimation and object instance segmentation-based semantic sparse map development.

In addition, a significant objective in intelligent robotics control is to enable robots to understand their environment to help us with various tasks [6, 7]. Specifically, the context of indoor search and rescue operations highlights the crucial role of UAVs in quickly scanning dangerous areas and providing real-time information to support emergency personnel. The success of such missions depends on the UAV's autonomous navigation capability to move through complex and cluttered real-world settings while ensuring safety. Typically, UAVs may encounter numerous obstacles and need to identify victims or hazardous zones, enabling immediate obstacle avoidance. This allows UAVs to discern obstacles and their locations, thus navigating through them. However, most real-world robots rely on 2D metric maps due to their simplicity and low resource requirements. In contrast, hollow bottom objects such as tables, desks, and chairs are often mapped as points or paths, leading to incorrect navigation decisions. This work addresses this issue by proposing a system that incorporates a 2.5D probabilistic semantic map, created by merging metric environmental structure with object height information, to enable UAVs to understand and interact with their surroundings more intelligently.

In general, in this thesis, we aim to achieve the following objectives:

- **To develop a Semantic SLAM system that combines metric mapping with semantic object information**: Our system will integrate geometric and semantic data to offer a more detailed comprehension of the environment.
- **To create an effective 3D map representation and provide a probabilistic semantic map**: We aim to enhance UAV navigation and mapping accuracy through detailed and probabilistic map representations.
- **To improve the accuracy and efficiency of UAV navigation and mapping in environments with hollow-bottom objects**: Our system will address the specific challenges posed by these types of objects.

- **To ensure the developed system is computationally efficient and suitable for deployment on UAVs with limited power and computational resources**: Ensuring efficiency is vital for real-time use and implementation on platforms with limited power and computational capacity.

1.3 Thesis Approach

To achieve these objectives, this thesis proposes a dual-approach Semantic SLAM system to address the problem of extracting and utilizing semantic information in UAV operations. The two approaches focus on 3D semantic mapping and 2.5D semantic mapping, respectively. The first approach focuses on 3D semantic mapping that merges cutting-edge visual SLAM to estimate a complete 6-DoF pose and sophisticated object segmentation techniques at the back end. To enhance the computational and storage efficiency of the framework, we adopt OctoMap, a streamlined voxel-based 3D map representation, to construct a functional system. In addition, we incorporate a fusion algorithm to acquire the semantic information from each frame in the SLAM task on the front end and the corresponding point. This integration of semantic data significantly improves the UAV's ability to perceive and navigate indoor environments, addressing challenges in pose estimation accuracy and uncertainty reduction. The second approach integrates a state-of-the-art Convolutional Neural Network (CNN) object detection framework at the front end and a 2D SLAM method - CartoGrapher [8] at the back end. We leverage the innovative BoT-SORT [9] methodology to effectively track and position semantic object classes extracted from the front-end interface [9] methodology. A novel association method is introduced to extract objects' positions and project them onto the metric map. In contrast to previous research, our approach focuses on ensuring reliable navigation in the environment with various hollow-bottom objects. The system output is a probabilistic map that significantly enhances the map's representation by incorporating object-specific attributes, including class distinctions, accurate positioning, and object heights. Finally, we evaluate our system by conducting comprehensive experiments using both publicly available datasets and simulation datasets.

1.4 Thesis Contributions

The main contribution of this thesis is to introduce a novel semantic SLAM framework to directly integrate high-level semantically significant entities (objects and structures) of the spare semantic SLAM system. By proposing a SLAM framework, we refer not only to the introduction of new mathematical representations suitable for merging semantics with visual SLAM, but also to leveraging object information in the front-end to develop new components for detecting and matching observations and hypothesizing constraints to create a more resilient topology for the underlying vSLAM.

The key contributions of this thesis are outlined as follows:

- **Proposing a 3D Semantic SLAM system**: We develop a system with faster 6-DoF pose tracking and the capability to construct a semantic sparse map based on object segmentation information.
- **Introducing an efficient representation and storage method using OctoMap**: This memory-efficient alternative to point cloud data enhances the front-end system
- **Implementing a 3D semantic mapping method**: Our method improves the extraction and utilization of semantic information in UAV operations.
- **Presenting a probabilistic metric map approach**: This approach incorporates localization and metric mapping with object tracking to enhance scene understanding, map accuracy, and UAV navigation capabilities.
- **Proposing a method to fuse extracted object information with a 2D map**: This creates a meaningful 2.5D map for obstacle avoidance.
- **Evaluating our SLAM system comprehensively**: We use public available SLAM benchmarks (TUM) and Gazebo simulation datasets. Additionally, we demonstrate the system's capability to construct semantically sparse maps in real-time on a compact, computation-limited platform via experiments on the Jetson Xavier AGX embedded computer.

1.5 Thesis Outline

The organization of this thesis is as outlined below:

- **Chapter 2:** In this chapter, we introduce an extensive review of the literature in the fields of SLAM, visual SLAM, and Semantic SLAM. This review covers the historical development and foundational

concepts of SLAM, discussing both traditional and modern approaches. We explore various methodologies for integrating semantic information into SLAM frameworks, highlighting key advancements and ongoing challenges. In addition, we examine cutting-edge techniques in object detection and object segmentation, which form the basis for the semantic understanding component of our proposed system.

- **Chapter 3:** This chapter provides an in-depth overview of the design and implementation of our proposed Semantic SLAM system. We begin with the theoretical underpinnings of SLAM and its extension to incorporate semantic data. The chapter elaborates on the system architecture, including the front-end and back-end components. We describe the integration of object segmentation and instance detection with traditional visual SLAM methods. Key innovations such as the association method and efficient mapping representation using OctoMap are discussed in detail. The chapter also covers the implementation aspects, including algorithms used for 6-DoF pose tracking, object detection, and map fusion techniques.
- **Chapter 4:** In this chapter, we outline the experimental setup employed to evaluate the performance of the proposed Semantic SLAM system. We describe the datasets employed, including both real-world and simulated environments, and the evaluation metrics used to assess system performance. The chapter presents an in-depth evaluation of experimental outcomes, contrasting our approach with existing state-of-the-art approaches. We evaluate the system's accuracy, computational efficiency, and robustness in different scenarios. Detailed insights into the advantages and disadvantages of the proposed system are provided based on empirical evidence.
- **Chapter 5:** This chapter concludes the thesis by summarizing the key findings and contributions of the research. We reflect on the implications of integrating semantic information into SLAM systems for UAV navigation and mapping. We also discuss potential future research directions, including enhancements to the semantic segmentation algorithms, real-time processing capabilities, and explore the use of this technology in broader fields like self-driving cars and industrial automation. Finally, we consider the feasibility of implementing the system on UAVs with constrained computational resources and suggest improvements for future iterations.

Chapter 2

Literature Review

In this chapter, we offer a comprehensive overview of the literature in the SLAM, visual SLAM, and Semantic SLAM fields. This review covers the historical development and foundational concepts of SLAM, discussing both traditional and modern approaches. We explore various methodologies for integrating semantic information into SLAM frameworks, highlighting key advancements and ongoing challenges. Additionally, we examine state-of-the-art techniques in object detection, image classification, and semantic segmentation, which form the basis for the semantic understanding component of our proposed system.

2.1 Fundamentals

2.1.1 3D Geometry

3D geometry is fundamental to various computer vision and robotics tasks, including SLAM, 3D reconstruction, and object recognition. This subsection outlines the essential concepts and equations that underpin 3D geometric transformations and representations. $SO(3)$, which denotes rotations, represents the special orthogonal group and can be expressed as:

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times3} \mid \mathbf{R}\mathbf{R}^T = 1,\, \det(\mathbf{R}) = 1 \right\} \tag{2.1}$$

The three-dimensional special Euclidean group $SE(3)$ can be described as:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \middle| \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{2.2}$$

where $\mathbf{R}$ denotes the 3×3 rotation matrix, $\mathbf{t}$ represents the translation vector, and $\mathbf{T}$ is the 4×4 transformation matrix, respectively.

Closure for $SE(3)$ from the first frame into the second frame can be seen simply by multiplying:

$$ {}^{1}\mathbf{T}_2 = \mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.3} $$

where, $\mathbf{R}_1\mathbf{R}_2 \in SO(3)$ and $\mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \in \mathbb{R}^3$.

The inverse of $SE(3)$ can be expressed as:

$$ {}^{2}\mathbf{T}_1 = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.4} $$

On the other hand, quaternions provide an alternative and effective method for depicting rotations in three-dimensional space. They avoid some of the pitfalls of rotation matrices, such as gimbal lock, and provide smoother interpolations. A quaternion $q = (x, y, z, w)^T$ with $w = \sqrt{1 - x^2 - y^2 - z^2}$, has an equivalent rotation matrix given by:

$$ \mathbf{R} = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2zw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix} \tag{2.5} $$

Bundle Adjustment

Bundle adjustment is an optimization technique used to refine the 3D coordinates of points and the parameters of the camera models to minimize the re-projection error. The projection error e for a point $\mathbf{p}^2_{c_j}$ observed in image i is given by:

$$ e^p_{ij} = \mathbf{p}^2_{c_{ij}} - \pi(\mathbf{R}_i, \mathbf{t}_i, \mathbf{p}^3_{c_j}) \tag{2.6} $$

where $\pi(\cdot)$ is the projection function

The objective is to reduce the total of squared re-projection errors to a minimum:

$$ \min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{p}^3_{c_j}} \sum_{i,j} ||e^p_{ij}||^2 \tag{2.7} $$

2.1.2 Mathematical Formulation of vSLAM

Visual SLAM (vSLAM) involves estimating the pose of the camera and constructing an environmental map using visual information. This procedure can be expressed as an optimization problem with the objective of minimizing the error between the observed and predicted feature positions. Non-linear least-squares optimization is frequently applied to address this issue.

State Representation and Observation Model

The state in vSLAM consists of the positions of the cameras and the three-dimensional coordinates of the observed landmarks. Let $\mathcal{X} \triangleq \{\mathbf{X}_t\}_{t=1}^T$

be the set of camera poses and $\mathcal{L} \triangleq \{\mathbf{l}_m\}_{m=1}^{M}$ be the set of 3D landmarks. Each landmark $\mathbf{l}_m$ is represented as:

$$\mathbf{l}_m = \begin{bmatrix} x_m & y_m & z_m \end{bmatrix}^T \tag{2.8}$$

The observation model $\mathcal{Z} \triangleq \{\mathbf{Z}\}$ describes the relationship between the 3D landmarks and their 2D projections in the image. Let $\mathbf{Z}_{ij}$ be the observation of the $j-th$ landmark in the $i-th$ camera frame. This relationship can be formulated as:

$$\mathbf{Z}_{ij} = \pi(\mathbf{R}_i, \mathbf{t}_i, \mathbf{l}_j) \tag{2.9}$$

Error Function

The error function measures the discrepancy between the observed feature positions and their predicted positions according to the current estimates of the camera poses and landmarks. The reprojection error e_{ij} for the observation $\mathbf{Z}_{ij}$ is given by:

$$e_{ij}^z = \mathbf{Z}_{ij} - \pi(\mathbf{R}_i, \mathbf{t}_i, \mathbf{l}_j) \tag{2.10}$$

The total error function E is the sum of squared reprojection errors for all observations:

$$E(\mathcal{X}, \mathcal{L}) = \sum_{i,j} ||e_{ij}^z||^2 \tag{2.11}$$

Non-linear Least-squares Optimization

The goal of non-linear least-squares optimization is to find the camera poses $\mathbf{X}$ and landmarks $\mathbf{l}$ that minimize the total error function E. This optimization problem is non-linear due to the projection function π. It can be solved using iterative methods such as Gauss-Newton or Levenberg-Marquardt algorithms.

The Gauss-Newton algorithm approximates the error function by linearizing the projection function around the current estimate. Let $\Delta\mathbf{X}_i$ and $\Delta\mathbf{l}_j$ be small increments to the camera pose $\mathbf{X}_i$ and landmark $\mathbf{j}_j$, respectively. The linearized error function is:

$$e_{ij} \approx e_{ij}^0 + \mathbf{J}_{ij} \begin{bmatrix} \Delta\mathbf{X}_i \\ \Delta\mathbf{l}_j \end{bmatrix} \tag{2.12}$$

where e_{ij}^0 is the initial error and $\mathbf{J}_ij$ is the Jacobian matrix of partial derivatives of the error with respect to the camera pose and landmark. The update step in the Gauss-Newton algorithm is obtained by solving the normal equations:

$$\mathbf{H}\Delta\theta = -g \tag{2.13}$$

where $\mathbf{H}$ is the Hessian matrix and g is the gradient of the error function:

$$\begin{aligned} \mathbf{H} &= \sum_{ij} \mathbf{J}_{ij}^T \mathbf{J}_{ij} \\ g &= \sum_{ij} \mathbf{J}_{ij}^T e_{ij}^0 \end{aligned} \tag{2.14}$$

The parameters are updated as:

$$\theta \leftarrow \theta + \Delta\theta \tag{2.15}$$

The Levenberg-Marquardt algorithm is an extension of the Gauss-Newton algorithm that incorporates a damping factor to ensure convergence. The update step is modified as:

$$(\mathbf{H} + \lambda\mathbf{I})\Delta\theta = -h \tag{2.16}$$

with θ representing the damping factor and $\mathbf{I}$ denoting the identity matrix. The damping factor λ is dynamically adjusted based on the reduction in the error function. If the error decreases, λ is decreased; otherwise, λ is increased.

Belief Inference

In the context of SLAM, belief inference refers to estimating the posterior distribution of the robot's state and the map given the observations. The belief $bel(\mathcal{X}, \mathcal{L})$ represents the probability distribution over the possible states of the camera poses and landmarks. The belief is updated using Bayes' theorem:

$$bel(\mathcal{X}, \mathcal{L}) = \eta p(\mathbf{Z}_t)|\mathcal{X}, \mathcal{L})bel(\mathcal{X}, \mathcal{L}) \tag{2.17}$$

The goal is to maximize the posterior probability:

$$\max_{\mathcal{X},\mathcal{L}} p(\mathcal{X}, \mathcal{L}|\mathbf{Z}_t) \tag{2.18}$$

which can be transformed into a minimization problem by taking the negative logarithm:

$$\min_{\mathcal{X},\mathcal{L}} -logp(\mathcal{X}, \mathcal{L}|\mathbf{Z}_t) \tag{2.19}$$

Assuming Gaussian noise, the negative log-likelihood is proportional to the sum of squared errors:

$$-logp(\mathcal{X}, \mathcal{L}|\mathbf{Z}_t) \propto \sum_t ||e_t^z||^2 \tag{2.20}$$

Thus, the optimization problem in belief inference aligns with the non-linear least-squares optimization problem, where the objective is to find the MAP (Maximum A Posteriori) estimate of the camera poses and landmarks. The optimization proceeds iteratively, updating the camera poses and landmarks until the change in the error function is below a predefined threshold or a maximum number of iterations is reached. The estimates of the camera poses and landmarks are refined through this process, resulting in a precise reconstruction of both the environment and the camera path.

2.1.3 Semantic Information

The term "*semantic* refers to the meaning or context of information within a given domain. In the context of SLAM, semantic information refers to the identification and categorization of objects and features within the environment, providing a richer understanding beyond a mere geometric representation. In our case, we call semantic information object-level data that enable a more comprehensive interpretation of the scene, such as recognizing furniture, distinguishing between different types of obstacles, and identifying points of interest.

Integrating semantic information into SLAM frameworks presents a significant challenge due to the intricacies involved in real-world environments and the need for accurate the efficient algorithms to process and understand the data. However, the development of deep learning, particularly CNN, has transformed the field by offering robust methods for object detection and object segmentation. These advances have enabled the incorporation of high-level semantic data into SLAM systems, enhancing their functionality and robustness.

Object Detection

Object detection entails the process of identifying and localizing objects within an image, providing essential information for semantic mapping. This process is fundamental for semantic mapping, as it provides the necessary information to label and categorize different elements in the environment. Early object detection methods relied on hand-crafted features and classifiers. The Deformable Part Model (DPM), proposed by Felzenszwalb *et al.* [10] (2008) was one of the most influential models, utilizing a combination of HOG (Histogram of Oriented Gradients) features and a linear SVM (Support Vector Machine) for object detection. Deep learning has revolutionized object detection, with CNNs becoming the benchmark because they can automatically learn hierarchical features from raw data. R-CNN (Region-based CNN) is introduced by Girshick *et al.* [11] (2014), R-CNN employs selective search to produce region proposals and a CNN to classify these regions.

Despite its high accuracy, R-CNN is computationally expensive as it requires running a CNN on each region proposal. Spatial Pyramid Pooling Network (SSP Net), proposed by He *et al.* [12] (2014), integrates a spatial pyramid pooling layer, which eliminates the fixed-size limitation of the input image, enabling the network to create fixed-length representations irrespective of the image's dimensions. While SSP Net improved detection accuracy, it also introduced computational complexity, which can be a limitation for real-time applications. Girshick [13] (2015) proposed Fast R-CNN, which improves speed by sharing convolutional features across proposals. Ren *et al.* [14] (2015) further optimized this approach with Faster R-CNN, introducing the Regional Proposal Network (RPN) that generates regional proposals within the network. However, these models continue to face challenges in real-time applications because of their comparatively high computational expense.

In addition, Redmon *et al.* [15] (2016) proposed YOLO, which formulates object detection as a unified regression task, estimating bounding boxes and class probabilities straight from the entire image. YOLO is known for its real-time performance and has seen several improvements, but initially struggled with small object detection and localization accuracy. Subsequent versions of YOLOv2 to YOLOv10 have addressed many of these issues, but can still be challenged by high-resolution images. Liu *et al.* [16] (2016) introduced SSD, which eliminates the need for region proposals by predicting bounding boxes and class scores for multiple default boxes of different aspect ratios at each location in several feature maps. SSD balances speed and accuracy effectively, though it can struggle with detecting small objects compared to region-based methods. FPN (Feature Pyramid Networks)is introduced by Lin *et al.* [17] (2017), which enhances object detection performance, especially for small objects, by constructing feature pyramids with lateral connections. The downside is the increased computational cost and complexity, which can affect real-time performance. Finally, He *et al.* [18] (2017) extended Faster R-CNN to include a branch for estimating segmentation masks within each region of interest, resulting in Mask R-CNN. This approach significantly improves instance segmentation accuracy but comes with higher computational demands, reducing its suitability for real-time applications.

Object Segmentation

Object segmentation entails labeling every pixel in an image with predefined categories, thereby creating detailed semantic maps that add to the robot's comprehension of its surroundings. Before deep learning, object segmentation methods relied on techniques such as graph cutting, conditional random fields (CRFs) [19], and manual feature engineering. These methods struggled with the complexity and variability of real-world scenes. On the other hand, deep learning has significantly advanced object segmentation.

CNNs have been adapted for pixel-wise classification, leading to several state-of-the-art models. Fully Convolutional Networks (FCN)is proposed by Long *et al.* [20] (2015), FCNs substitute fully connected layers in traditional CNNs with convolutional layers, allowing the network to output spatial maps instead of class scores for end-to-end segmentation training. However, FCNs can suffer from coarse segmentation outputs due to downsampling in the network. Badrinarayanan *et al.* [21] (2017) introduced SegNet, using an encoder-decoder architecture where the encoder extracts features and the decoder upsamples the feature maps to produce the final segmentation. SegNet is known for its efficient memory usage, but it can be less accurate than other deep learning models. Ronneberger *et al.* [22] (2015) developed U-Net for biomedical image segmentation. It uses an encoder-decoder architecture with skip connections linking corresponding layers, preserving spatial information and improving accuracy. U-Net is highly effective but can be computationally intensive due to its complex architecture.

Chen *et al.* [23] (2018) introduced DeepLab, incorporating atrous (dilated) convolutions for multi-scale context and Conditional Random Fields (CRFs) for precise boundary localization. DeepLab has evolved through versions, including DeepLabv2 and DeepLabv3. While DeepLab achieves high accuracy, it requires significant computational resources. Pohlen *et al.* [24] (2017) introduced FRRN (Full Resolution Residual Networks), which combines multi-scale context aggregation with residual connections to maintain high-resolution information throughout the network. FRRN is effective for semantic segmentation, but it can be computationally intensive. ICNet (Image Cascade Network) is introduced by Zhao *et al.* [25] (2018), designed for real-time semantic segmentation by cascading feature maps of different resolutions. ICNet achieves a good balance between accuracy and speed, but can still be less accurate than more complex models. PSPNet (Pyramid Scene Parsing Network) is proposed by Zhao *et al.* [26] (2017), PSPNet uses pyramid pooling to capture global context information, enhancing segmentation performance in complex scenes. PSPNet delivers state-of-the-art results, but at the cost of increased computational demand.

2.2 Related Works

The evolution of vSLAM has been marked by significant advancements in algorithms, sensor integration, and computational efficiency. This section reviews the progression of vSLAM techniques, from early methods to state-of-the-art approaches, highlighting key developments and their contributions to the field.

2.2.1 Visual SLAM Evolution

Early vSLAM methods focused on feature extraction and matching to determine the camera's pose and construct a map of the surroundings. These techniques relied heavily on handcrafted features and geometric transformations. Among the earliest real-time vSLAM systems using a single camera was MonoSLAM, introduced by Davison *et al.* [27] in 2007. MonoSLAM used an Extended Kalman Filter (EKF) to estimate the camera trajectory and map the environment. Around the same time, Klein and Murray [28] proposed PTAM (Parallel Tracking and Mapping), which separated tracking and mapping into parallel processes, improving the system's robustness and real-time performance.

Feature-based methods soon became popular due to their robustness in various environments. These methods detect and match key points across frames to estimate motion and build maps. In 2015, Mur-Artal *et al.* [29] introduced ORB-SLAM, which uses ORB (Oriented FAST and Rotated BRIEF) features for tracking, mapping, and loop closure detection. ORB-SLAM is highly efficient and works well in large-scale environments. Another significant contribution was LSD-SLAM [30] (Large-Scale Direct SLAM), developed by Engel *et al.* in 2014. LSD-SLAM performs direct image alignment without explicit feature extraction, allowing for dense 3D reconstruction. More recently, Wu *et al.* [31] (2020) introduced CubeSLAM, a feature-based SLAM system that integrates object detection with traditional feature-based SLAM to improve robustness in complex environments by leveraging the geometric information of cuboid shapes.

Direct methods, in contrast to feature-based methods, use the intensity information of the pixels directly to estimate the camera motion and build maps. Forster *et al.* [32] introduced SVO (Semi-Direct Visual Odometry) in 2014, which combines the efficiency of direct methods with the robustness of feature-based methods, optimizing the pose using both direct and indirect (feature-based) measurements. Engel *et al.* [33] proposed DSO (Direct Sparse Odometry) in 2017, a direct method that optimizes photometric error over a sparse set of pixels, providing high-precision camera tracking. The integration of deep learning into vSLAM has led to significant improvements in accuracy and robustness, particularly in dynamic and unstructured environments. Wang *et al.* [34] introduced DeepVO (Deep Visual Odometry) in 2017, leveraging recurrent neural networks (RNNs) to learn motion patterns from sequences of images, providing robust pose estimation in challenging conditions. Teed and Deng developed DeepV2D [35] in 2018, a deep learning-based approach that integrates differentiable rendering with pose estimation to achieve high accuracy in dynamic and challenging environments. Czarnowski

et al. [36] proposed DeepFactors in 2020, which combines traditional factor graph optimization with learned dense feature representations to improve robustness and accuracy in vSLAM.

Recent advancements in vSLAM have focused on improving robustness, efficiency, and the ability to handle dynamic environments. Qin *et al.* [37] proposed VINS-Mono in 2018, a tightly-coupled visual-inertial odometry system that integrates IMU data with visual information to improve robustness and accuracy, particularly in scenarios with rapid motion. Schops *et al.* [38] proposed BAD-SLAM in 2019, combining bundle adjustment with deep learning features for dense SLAM, improving the accuracy and efficiency of 3D reconstruction. Mur-Artal *et al.* [39] extended ORB-SLAM to support monocular, stereo, and RGB-D cameras with ORB-SLAM3 in 2021, introducing improved tracking, mapping, and loop closure capabilities, making it one of the most versatile vSLAM systems available. Teed and Deng [40] introduced DROID-SLAM in 2021, which employs recurrent neural networks to perform both dense and sparse visual SLAM, achieving state-of-the-art performance in various benchmarks.

The advent of new sensors, such as RGB-D, stereo, and event cameras, has further advanced the field of vSLAM by providing richer and more diverse data for mapping and localization.

- **RGB-D Cameras:** These cameras provide synchronized RGB and depth data, enabling more accurate and detailed environmental mapping. Examples include ElasticFusion (2015) by Whelan *et al.* [41], which uses a surfel-based fusion method for dense SLAM, and ORB-SLAM2 (2017) by Mur-Artal *et al.* [42], which extends ORB-SLAM to support RGB-D cameras.
- **Stereo Cameras:** These cameras use two lenses to capture 3D information through the disparity between images. Stereo ORB-SLAM (2017) by Mur-Artal *et al.* [42] and VINS-Fusion (2018) by Qin *et al.* [43] are notable examples that leverage stereo vision for robust SLAM.

The following table 2.1 summarizes and compares various traditional vSLAM approaches based on their key attributes and performance metrics.

Table 2.1: A comparison between traditional vSLAM approaches: M - Mono Camera, R - RGBD Camera, S - Stereo Camera, B - Bundle Adjustment, DL - Deep Learning, H - Hybrid, F - Feature-based method, D - Direct method, S - Single-threaded, Mu - Multi-threaded, Sta - Static, Dyn - Dynamic

SLAM	Sensors	Method	Component	Architecture	Library	Env. Rep.	Map Den.	Env. Struct.	Large Env.	Dyn. Env.	Loop Clos.	Online	Pub. Avai.
MonoSLAM [27]	M	EKF	F	S	No	Sparse	Low	Sta	No	No	Yes	Yes	No
PTAM [28]	M	B	F	Mu	No	Sparse	Low	Sta	No	No	Yes	Yes	No
LSD-SLAM [30]	M	D	D	Mu	Yes	Dense	High	Sta	Yes	No	Yes	Yes	Yes
ORB-SLAM [29]	M	ORB	F	Mu	Yes	Sparse	High	Sta	Yes	No	Yes	Yes	Yes
DSO [33]	M	D	D	Mu	Yes	Sparse	High	Sta	Yes	No	No	Yes	Yes
DeepVO [34]	M	DL	DL	S	No	Sparse	High	Dyn	Yes	Yes	No	Yes	No
VINS-Mono [37]	M, IMU	F	H	Mu	Yes	Sparse	High	Dyn	Yes	Yes	Yes	Yes	Yes
DeepFactors [36]	M	DL	H	Mu	No	Dense	High	Dyn	Yes	Yes	Yes	Yes	No
DeepV2D [35]	M	DL	DL	Mu	No	Dense	High	Dyn	Yes	Yes	Yes	Yes	No
ORB-SLAM3 [39]	M,S,R	ORB	F	Mu	Yes	Sparse	High	Dyn	Yes	Yes	Yes	Yes	Yes
CubeSLAM [31]	M	F	F	Mu	Yes	Sparse	High	Dyn	Yes	No	Yes	Yes	Yes
DROID-SLAM [40]	M	DL	H	Mu	No	Dense	High	Dyn	Yes	Yes	Yes	Yes	No
BAD-SLAM [38]	M	DL	H	Mu	No	Dense	High	Dyn	Yes	Yes	Yes	Yes	No
VINS-Fusion [43]	S, IMU	F	H	Mu	Yes	Sparse	High	Dyn	Yes	Yes	Yes	Yes	Yes
ElasticFusion [41]	R	Surfel	D	Mu	Yes	Dense	High	Sta	Yes	No	Yes	Yes	Yes

2.2.2 From Traditonal SLAM to Semantic SLAM

The transition from traditional SLAM to Semantic SLAM represents a significant advancement in robotic perception and navigation. Traditional SLAM systems focus primarily on geometric reconstruction and pose estimation, often lacking contextual comprehension of the surroundings. Conversely, Semantic SLAM, integrates high-level semantic information, such as object recognition and scene understanding, into the SLAM framework. Traditional SLAM systems are designed to estimate the robot's pose and construct a map of the environment simultaneously. These systems rely heavily on feature extraction, matching, and geometric transformations. While they are effective in building accurate maps and tracking the robot's movement, they do not provide details regarding the categories and identities of items in the surroundings.

Semantic SLAM builds upon traditional SLAM by incorporating semantic information into the mapping and localization process. This integration enhances the robot's understanding of its surroundings, enabling it to recognize and categorize objects, understand their relationships, and perform higher-level tasks. Frank Dellaert and David Bruemmer [44] were pioneers in investigating and interpreting the issue of semantic mapping. Recent advancements in deep learning technologies have empowered researchers to tackle the conventional SLAM problem. By employing deep learning techniques, researchers are able to extract feature points, generate descriptors, obtain semantic information, and estimate poses. Integrating semantic information into standard vSLAM improves the understanding of image features and results in highly accurate semantic maps. [45–47]. When the aim is to enhance localization, mapping, or both, these challenges are termed semantic localization, semantic mapping, and semantic SLAM, respectively. SemanticFusion, introduced by McCormac *et al.* [46] in 2017, combines dense RGB-D SLAM with real-time semantic segmentation to create a semantically annotated 3D map. This approach leverages deep learning to perform pixel-wise segmentation, integrating the results into the SLAM pipeline. Similarly, Co-Fusion [48], Mask Fusion [49], DS-SLAM [50], and SCFusion [51], extending the SLAM system with instance-level segmentation using Mask R-CNN. These methods not only reconstructs the 3D environment but also identifies and labels individual objects, providing a detailed semantic map.

Current research efforts have focused mainly on creating 3D semantic maps using technologies such as depth cameras [6], [52], stereo cameras [53], 3D LiDAR [54], or fusion sensors [55]. However, these methods frequently require significant computational power and storage, rendering them less appropriate for UAVs with strict weight and computational limitations and

complicating their use during UAV navigation. Various studies have also applied semantic mapping for tasks like room classification [56] and dynamic object detection [57]. However, most practical robots use 2D metric maps due to their simplicity and low resource requirements. Furthermore, the mapping of hollow bottom objects like tables, desks, and chairs are usually mapped on the map as points or paths leading to wrong navigation decisions. Further advancements include Fusion++, proposed by McCormac *et al.* [58] in 2018, which integrates semantic segmentation and object recognition into a dense SLAM system. Fusion++ uses CNNs to segment images and fuse the semantic labels into the 3D map, enhancing the robot's environmental understanding. These developments have greatly enhanced the precision and reliability of SLAM systems, enabling them to handle complex dynamic environments more effectively.

Despite these advancements, Semantic SLAM faces several challenges, including computational complexity, real-time performance, and robustness in dynamic environments. Future research focuses on overcoming these obstacles by creating more efficient algorithms, leveraging advanced machine learning techniques, and improving sensor integration. Enhancing the capabilities of Semantic SLAM systems will enable more sophisticated and autonomous robotic applications, driving further innovation in the field. The evolution from traditional SLAM to Semantic SLAM represents a paradigm shift in robotic perception. By integrating semantic information, these systems provide a richer, more comprehensive understanding of the surroundings, allowing robots to execute complex tasks with greater autonomy and intelligence. The advancements in deep learning and sensor technology continue to drive this field forward, promising exciting developments in the future.

In this thesis, our aim is to advance the field of Semantic SLAM by developing a system that combines metric mapping with semantic object information. Our approach involves improving the precision and effectiveness of UAV navigation and mapping in environments with hollow-bottom objects, ensuring computational efficiency suitable for deployment on UAVs with limited power and computational resources. By integrating state-of-the-art object detection and segmentation techniques with robust SLAM algorithms, our system aspires to achieve real-time, semantically-rich mapping capabilities.

Chapter 3

Methodology

This chapter presents the detailed design and implementation of the proposed Semantic SLAM system. We begin with the theoretical underpinnings of vSLAM and its extension to incorporate semantic data. The chapter elaborates on the system architecture, including the front-end and back-end components. We describe the integration of semantic segmentation and object detection with traditional vSLAM methods. Key innovations such as the association method and efficient mapping representation using OctoMap are discussed in detail. The chapter also covers the implementation aspects, including algorithms used for 6-DoF pose tracking, object detection, and map fusion techniques.

3.1 System Overview

The proposed semantic SLAM is illustrated in Fig. 3.1, which includes three main components: Localization modules, 3D Semantic Map creator module and 2.5D Semantic Map creator module. Our system utilizes RGB color information and depth information from a single depth camera. The Localization module is responsible for tracking, local mapping, and loop closing. Tracking ensures continuous pose estimation; local mapping creates a local representation of the environment, and loop closing corrects drifts by recognizing previously visited locations. The 2.5D Semantic Map creator focuses on integrating semantic information into a 2D map. This process involves object positioning, where detected objects are located within the environment, and projection, which maps these objects onto a 2D plane. The module then creates a 2.5D semantic probability map by combining the projected data with metric mapping, resulting in a detailed 2D map with semantic information. The 3D Semantic Map creator performs more complex operations to generate a 3D map. It begins with semantic segmentation, which categorizes different regions of the RGB-D images. Semantic point cloud generation then creates a 3D point cloud with semantic labels. This point cloud undergoes semantic fusion, integrating multiple observations to

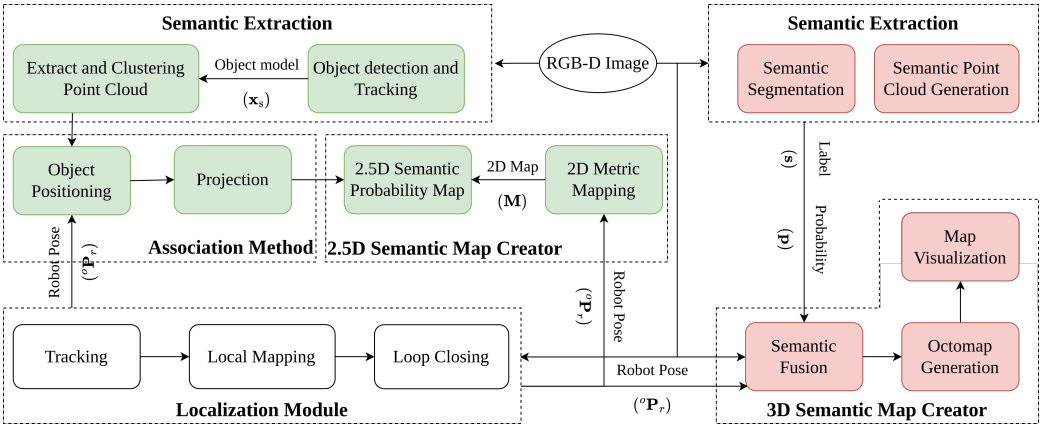


Figure 3.1: **Proposed Semantic SLAM Architecture**: The system is composed of three units: a full 6 DoF pose estimation of the drone, a 3D semantic mapping branch, and a 2.5D semantic mapping branch.

build a consistent model. The final step, Octomap generations, constructs a spare map that represents the 3D structure of the environment, The resulting 3D maps is then visualized for interpretation and analysis. This architecture ensures robust and precise environmental understanding by combining real-time localization with detailed 2D and 3D semantic mapping, enhancing the capabilities of autonomous systems in complex environments.

3.2 Visual Localization

Visual Localization is a crucial component of our proposed Semantic SLAM system, enabling UAVs to navigate and map their environment using visual information. This section delves into the theoretical foundations of visual localization, highlighting the key mathematical concepts and equations that underpin this technology.

3.2.1 Pose Estimation

Pose estimation in our system involves determining the position and orientation of the UAV in 6 DoF (degrees of freedom) from visual input. This can be formulated as an optimization problem, where the objective is to minimize the reprojection error between the observed features in the image and their corresponding 3D points in the map.

We utilize the ORB-SLAM3 algorithm [39] for accurate and real-time estimation of camera poses from RGB-D images. ORB-SLAM3 leverages a

monocular camera model, extending its capabilities to support both stereo and RGB-D configurations, making it idea for our UAV's sensor setup. It include three concurrent parts: (1) Tracking, (2) Local Mapping, and (3) Loop Closing [59]. The challenge of estimating pose involves determining the UAV's position (x, y, z) and orientation (ϕ, θ, ψ) in a global frame of reference. ORB-SLAM3 addresses this by tracking a collection of distinctive features across successive frames and establishing the correspondences among them. The estimated pose is derived by minimizing the re-projection error between the observed feature positions and their predicted positions in the camera frame. Mathematically, given a set of N observed 2D feature points $\mathbf{p}_i^2$ in the current RGB-D frame and their corresponding 3D points $\mathbf{p}_i^3$ in the world frame, the estimated camera pose ${}^{\bar{o}}\mathbf{T}_c \in SE(3)$ with:

$${}^{\bar{o}}\mathbf{T}_c = \begin{bmatrix} {}^{\bar{o}}\mathbf{R}_c & {}^{\bar{o}}\mathbf{t}_c \\ 0_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} {}^{\bar{o}}r_{c_{00}} & {}^{\bar{o}}r_{c_{01}} & {}^{\bar{o}}r_{c_{02}} & {}^{\bar{o}}t_{c_{00}} \\ {}^{\bar{o}}r_{c_{10}} & {}^{\bar{o}}r_{c_{11}} & {}^{\bar{o}}r_{c_{12}} & {}^{\bar{o}}t_{c_{10}} \\ {}^{\bar{o}}r_{c_{22}} & {}^{\bar{o}}r_{c_{22}} & {}^{\bar{o}}r_{c_{22}} & {}^{\bar{o}}t_{c_{20}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

can be obtained by solving the optimization problem:

$${}^{\bar{o}}\mathbf{T}_c = \underset{{}^{\bar{o}}\mathbf{T}_c}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{p}_i^2 - \pi({}^{\bar{o}}\mathbf{T}_c \times \mathbf{p}_i^3)||^2 \tag{3.1}$$

where ${}^{\bar{o}}\mathbf{R}_c, {}^{\bar{o}}\,\mathbf{t}_c, {}^{\bar{o}}\,\mathbf{T}_c$ represent the rotation matrix, the translation matrix, and the transformation matrix, respectively, between the world coordinate frame ($\bar{\mathbf{O}}_{xyz}$) and the camera coordinate frame. The function $\pi(\cdot)$ is the projection function from 3D to 2D points and $||\cdot||$ indicates the Euclidean distance.

Since camera odometry obtained from Eq. 3.1 and robot odometry utilize different world coordinates, a calibration process was carried out. Let ${}^{o}\mathbf{T}_r, {}^{o}\,\mathbf{T}_c$ denote the transformation matrix representing the robot pose and camera pose relative to the robot's world frame ($\mathbf{O}_{xyz}$), respectively. The transformation ${}^{o}\mathbf{T}_c$ is consequently calculated as:

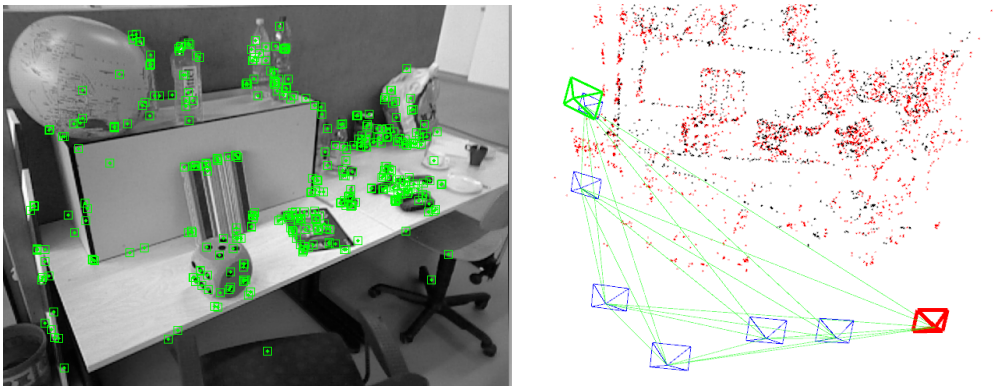
$${}^{o}\mathbf{T}_c = \begin{bmatrix} {}^{\bar{o}}r_{c_{00}} & {}^{\bar{o}}r_{c_{02}} & {}^{\bar{o}}r_{c_{21}} & {}^{\bar{o}}t_{c_{20}} \\ {}^{\bar{o}}r_{c_{20}} & {}^{\bar{o}}r_{c_{11}} & {}^{\bar{o}}r_{c_{01}} & {}^{\bar{o}}t_{c_{00}} \\ {}^{\bar{o}}r_{c_{12}} & {}^{\bar{o}}r_{c_{10}} & {}^{\bar{o}}r_{c_{22}} & {}^{\bar{o}}t_{c_{10}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Consider ${}^{c}\mathbf{T}_r$ as the transformation matrix from the camera frame to the UAV frame. The pose of the UAV is expressed by:

$${}^{o}\mathbf{T}_r = {}^{o}\,\mathbf{T}_c \times {}^{c}\,\mathbf{T}_r \tag{3.3}$$

3.2.2 Feature Matching

Feature matching is essential for establishing correspondences between successive frames. Common feature descriptors such as SIFT, SURF, or ORB are used to detect and describe key points in images. The matching process can be formulated as finding pairs of key points $(\mathbf{p}_i, \mathbf{p}_j)$ between frames I_i and I_j that satisfy certain similarity criteria. Illustrated in Fig. 3.2 are the outcomes of feature extraction and pose tracking.



(a) ORB Feature Extraction (b) UAV Pose Tracking

Figure 3.2: An Example of feature extraction and pose tracking.

3.2.3 Motion Model

The UAV's motion can be described by a kinematic model. Assuming a constant velocity model, the motion between two time steps can be represented as:

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t, \mathbf{u}_t) + \mathbf{w}_t \tag{3.4}$$

where, $\mathbf{X}_t$ is the UAV reference state matrix at time t,, $\mathbf{u}_t$ is the control input, $\mathbf{w}_t$ is the process noise. $\mathbf{X}_t$ typically include the UAV's position and orientation $\begin{bmatrix} x_t & y_t & z_t & \theta_t & \phi_t & \psi_t. \end{bmatrix}$

In our system, motion control is managed through waypoint navigation. This approach involves defining a set of goals or waypoint that the UAV must reach, which allows for structured and efficient path planning. A waypoint $\mathbb{W}_k \in SE(3)$ can be defined as a position and orientation in the 3D space. The control objective is to navigate the UAV from its current state $\mathbf{X}_t$ to the next waypoint $\mathbf{W}_{k=1}$. The control law can be designed using a Proportional-Intergral (PI) controller and a $SO(3)$ control for orientation. The PID control law is formulated as follows:

$$\mathbf{u}_t = k_p \mathbf{e}_t + k_i \int_0^t \mathbf{e}_\tau d_\tau \tag{3.5}$$

where $\mathbf{e}_t = \begin{bmatrix} x_{k+1}^W - x_t & y_{k+1}^W - y_t & z_{k+1}^W - z_t \end{bmatrix}^T$ is the error between the desired waypoint and the current position, k_d and k_i are proportional and integral gains, respectively. $\int_0^T \mathbf{e}_\tau d_\tau$ is the integral of the error over time. The PI controller adjusts the UAV's control input $\mathbf{u}_t$ to minimize the error $\mathbf{e}_t$, ensuring that the UAV follows the desired trajectory.

The orientation of the UAV is controlled using an $SO(3)$ controller to ensure smooth and accurate rotations. The special orthogonal $SO(3)$ represents the space of all possible rotations in three dimensions. Given a desired orientation $\mathbf{R}_d \in SO(3)$ and the current orientation $\mathbf{R} \in SO(3)$, the error in orientation $\mathbf{e}_R$ can be defined as:

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_d^T \mathbf{R} - \mathbf{R}^T \mathbf{R}_d)^\vee \tag{3.6}$$

where $(\cdot)^\vee$ denotes the vee operation, which maps a skew-symmetric matrix to a vector in $\mathbb{R}^3$

The control law for the orientation is given by the following:

$$\mathbf{u}_R = k_r \mathbf{e}_R + k_\Omega (\Omega_d - \Omega) \tag{3.7}$$

where k_R and k_Ω are the orientation and angular velocity gains, respectively. Ω_d is the desired angular velocity and Ω is the current angular velocity.

The integrated control system leverages the PI controller for position and the $SO(3)$ controller for orientation to navigate the UAV through a sequence of waypoints. The overall path can be represented as a sequence of waypoints $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N\}$, with each waypoint specifying both position and orientation goals: $\mathbf{W}_k = \big([x_k, y_k, z_k]^T, \mathbf{R}_k\big)$.

3.2.4 Loop Closure

Loop closure is a critical component of vSLAM, helping to correct drift in the UAV's trajectory by recognizing previously visited locations. When a loop closure is detected, a constraint is added between the current pose and the pose corresponding to the previously visited location. The formulation of loop closure can be represent as: $\mathbf{Z}$

$$\mathbf{X}_t = g(\mathbf{X}_k, \mathbf{Z}_{tk}) + \mathbf{w}_{tk}^m \tag{3.8}$$

where g is a function that transforms the pose $\mathbf{X}_k$ to $\mathbf{X}_t$ based on the observed loop closure measurement $\mathbf{Z}_{tk}$ and $\mathbf{w}_{tk}^m$ is the measurement noise.

The optimization problem with loop closure can be expressed as:

$$\min_{\mathbf{X},\mathbf{L}} \sum_{i} \sum_{j} ||\mathbf{x}_{ij} - \mathbf{K}(\mathbf{R}_i\mathbf{L}_j + \mathbf{t}_i)||^2 + \sum_{(t,k} ||\mathbf{X}_t - g(\mathbf{X}_k, \mathbf{Z}_{tk})||^2 \tag{3.9}$$

where, $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{t}$ is translation vector, and $\mathbf{x}_{ij}$ is the observed image point corresponding to landmark $\mathbf{L}_j$ in pose $\mathbf{X}_i$.

3.2.5 Probabilistic Formulation

In this thesis, we formulate this problem as a probabilistic perspective, where the objective is to estimate the posterior distribution of the UAV's trajectory and map given the observations. Using Bayes' theorem, the posterior distribution is:

$$p(\mathcal{X}, \mathcal{L}|\mathcal{Z},\mathcal{U}) \propto p(\mathcal{Z}|\mathcal{X}, \mathcal{L})p(\mathcal{X}|\mathcal{U})p(\mathcal{L}) \tag{3.10}$$

where $\mathcal{U}$ denotes set of all control input, $p(\mathcal{Z}|\mathcal{X}, \mathcal{L})$ is the likelihood of the observations given the trajectory and map, $p(\mathcal{X}|\mathcal{U})$ is the prior distribution of the trajectory given the control inputs, and $p(\mathcal{L})$ is the prior distribution of the map. The estimation process involves maximizing the posterior distribution, we implemented using Graph-Based Optimization [60] technique. By grounding our Semantic SLAM system in these theoretical underpinnings of vSLAM, we leverage robust techniques for pose estimation, feature matching, motion modeling, map representation, and loop closure detection. These components work together to enable accurate and efficient mapping and navigation for UAVs in complex environments.

3.3 3D Semantic SLAM with Object Segmentation

The proposed 3D semantic SLAM methodology takes RGB-D sequences as input and incrementally constructs a volumetric map augmented with object instances. Initially, RGB-D images undergo preprocessing through a UAV pose tracking framework (Section 3.2). Subsequently, an object segmentation method is employed to identify and extract semantic 3D objects from individual frames (Section 3.3.1). These extracted objects are then associated with a volumetric mapping framework to generate a dense object-level map (Section 3.3.3). To further refine map quality, Octomap is utilized for noise attenuation and voxel grid downsampling to conserve spatial resources, and optimization for enhance visual representation (Section 3.3.4).

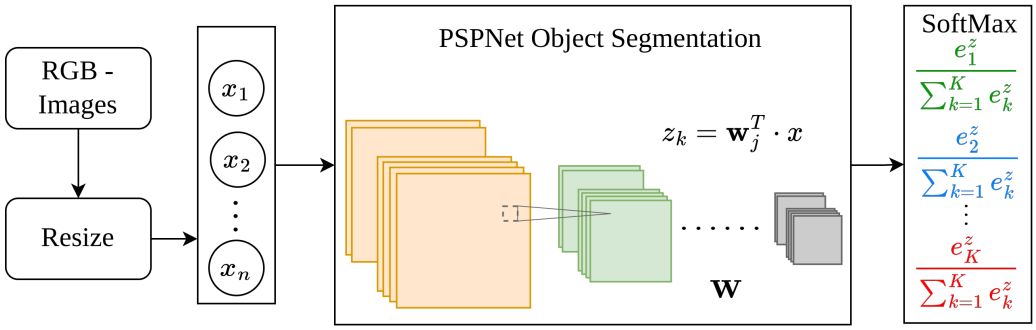


Figure 3.3: Structure of semantic segmentation model

The system is implemented within the ROS framework, a widely adopted platform in the robotics domain, leveraging open-source tools, libraries, and interoperability to facilitate the development of complex and robust robotic behaviors.

3.3.1 Semantic segmentation

Semantic segmentation is a crucial component in our methodology, responsible for extracting meaningful object instances from RGB-D images. This section elaborates on the process, utilizing the Pyramid Scene Parsing Network (PSPNet) [26] for its robust performance in semantic segmentation tasks, which is illustrated in Fig. 3.3. The input to the semantic segmentation module consists of color images, where each image undergoes an initial resizing step to fit the input dimensions required by the PSPNet. The resizing process ensures that the images are compatible with the network architecture, which is essential for maintaining the accuracy and efficiency of the segmentation process. Let $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ represent an RGB image of height H and with W. The image is resized to the input size $\mathbf{I}' \in \mathbb{R}^{H' \times W' \times 3}$, where H' and W' are the height an width dimensions specified by the PSPNet.

PSPNet employs a deep convolutional neural network, ResNet, for feature extraction. ResNet processes the resized image $\mathbf{I}'$ to extract high-level features, resulting in a feature map $\mathbf{F} \in \mathbb{R}^{H' \times W' \times C}$, where C is the number of channels in the feature map. The feature extraction process can be mathematically expressed as:

$$\mathbf{F} = ResNet(\mathbf{I}') \tag{3.11}$$

The core innovation of PSPNet lies in its pyramid pooling module, which captures contextual information at multiple scales. The feature map $\mathbf{F}$ is divided into several sub-regions, and average pooling is applied to each region.

This process generates pooled feature maps at different scales represented as $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4\}$, where each $\mathbf{P}_i \in \mathcal{R}^{h_i \times w_i \times C}$ corresponds to a different pyramid level. The pooled features are then upsampled to the original feature map size:

$$\mathbf{P}'_i = Upsample(\mathbf{P}_i) \tag{3.12}$$

The upsampled pooled feature maps are concatenated with the original feature map $\mathbf{F}$, resulting in a fused feature map $\mathbf{F}'$:

$$\mathbf{F}' = Concat(\mathbf{F}, \mathbf{P}'_1, \mathbf{P}'_2, \mathbf{P}'_3, \mathbf{P}'_4) \tag{3.13}$$

The fused feature map $\mathbf{F}'$ undergoes additional convolutional layers to refine the features and generate a class score map $\mathbf{S} \in \mathcal{R}^{H' \times W' \times K}$, where K is the number of semantic classes. The class score for each pixel is computed as:

$$\mathbf{S}_{ijk} = (\mathbf{W}^e_k)^T \mathbf{F}_{ij} + \mathbf{b}_k \tag{3.14}$$

where $\mathbf{W}^e_k$ and $\mathbf{b}_k$ are the weights and biases associated with class k.

To obtain a probability distribution over the classes for each pixel, a softmax activation function is applied to the class score map $\mathbf{S}$:

$$p_{ijk} = \frac{e^{\mathbf{S}_{ijk}}}{\sum_{k'=1}^{K} e^{\mathbf{S}_{ijk'}}} \tag{3.15}$$

Each pixel, along with its probability, is then selected and fused with the point cloud's pose.

3.3.2 Pointcloud generation and Semantic Pointcloud structure

The process begins with the acquisition of RGB-D images from the UAV's onboard sensors. These images provide both color (RGB) and depth information, crucial for generating a 3D point cloud. Each pixel in the depth image corresponds to a point in the 3D space, with its coordinates calculated using the camera's intrinsic parameters. The transformation from a 2D pixel (u, v) in the image plane to a 3D point $(X, Y, Z))$ in the camera coordinate system is given by

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & f_y \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ d(u, v) \end{bmatrix} \tag{3.16}$$

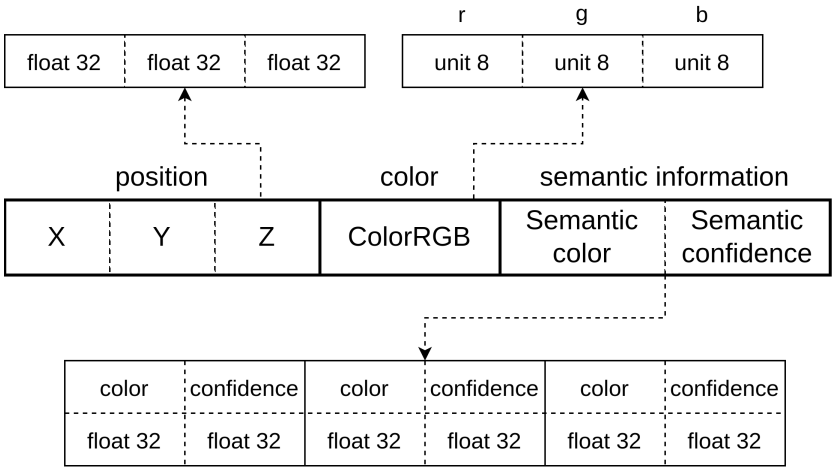


Figure 3.4: Illustration of semantic point cloud structure

where c_x, c_y) are the principal point coordinates, (f_x, f_y) are the focal lengths, and $d(u, v)$ is the depth value at pixel (u, v). By iterating through each pixel, a dense point cloud is generated that represents the 3D structure of the environment. This point cloud serves as the foundational layer for subsequent semantic processing.

The semantic point cloud structure builds upon the basic point cloud by incorporating semantic labels obtained from the semantic segmentation process. Each point in the cloud is enriched with semantic information, allowing the SLAM system to not only map the environment geometrically but also understand the types of objects present within it. Once the semantic labels are obtained, they are merged with the 3D point cloud. Each point $\mathbf{p}_i^3$ in the cloud is now associated with semantic labels s_{i1}, s_{i2}, s_{i3} and their corresponding confidence scores p_{i1}, p_{i2}, p_{i3} resulting in a semantic point cloud $\{(\mathbf{p}_i^3, (s_{i1}, p_{i1}), (s_{i2}, p_{i2}), (s_{i3}, p_{i3}))\}_{i=1}^N$. This semantic enrichment enables the system to differentiate between various objects, such as furniture, walls, and other obstacles, within the mapped environment. The structure of the semantic point cloud is illustrated in Figure 3.4. Each point in the semantic point cloud is represented with the following attributes:

- X, Y, Z coordinates (*float32*): 3D spatial information.
- RGB color values (*unit8*): Color information from the RGB image.
- Semantic colors (*unit8*): Three top class labels assigned by the semantic segmentation network.
- Semantic confidence (*float32*): Confidence scores of the three assigned labels.

This structured representation ensures that the semantic point cloud contains comprehensive information about the environment, combining geometric, color, and semantic data.

3.3.3 Semanic fusion

To enhance the quality and robustness of the semantic point cloud, the semantic information is fused across multiple views. As the UAV moves and captures new RGB-D frames, each frame's semantic point cloud is integrated into a global point cloud. This fusion process helps in refining the semantic labels and confidence scores by considering the consistency of labels across different views. This data is characterized by the vector $\mathbf{Q} = \begin{bmatrix} \mathbf{t} & c & \mathbf{s} & \mathbf{p} \end{bmatrix}^T$, where $\mathbf{t} \in \mathbb{R}^3$ represents the 3D spatial coordinates, and $c \in \mathbb{R}^1$ corresponds to the RGB color of the point cloud. In addition, $\mathbf{s} \in \mathbb{R}^k$ and $\mathbf{p} \in \mathbb{R}^k$ indicate the k most probable semantic colors and their respective confidence scores associated with a point cloud.

For each observation O_i, we determine the probability of every semantic color within a specified semantic set. Then, the point with the highest probability is chosen as the final decision. This method guarantees that semantic information is thoroughly integrated into the point cloud, allowing for a more detailed and nuanced comprehension of the scene across various viewpoints and translations. The algorithm 3.1 illustrates the semantic fusion procedure.

Following the above process, our proposed semantic SLAM system effectively integrates semantic information across multiple views, enhancing the richness and accuracy of the generated semantic point cloud. This integration not only improves the understanding of the scene but also aids in better decision-making for UAV navigation and interaction within complex environments.

3.3.4 Semantic map creation

In our method, each keyframe stores the 3D point clouds, while the segmented point clouds are maintained corresponding to their respective objects. However, point cloud-based maps typically require significant storage capacity, making them impractical for large-scale environment modeling with limited memory. Moreover, the absence of efficient structures to store each point complicates search operations. Moreover, they lack structures to efficiently store each point, hindering search operations, and failing to provide volume information for individual points, limiting their usefulness for advanced tasks like path planning or grasp point selection.

Algorithm 3.1 Semantic Fusion Approach

Input: $\mathbf{Q}_1$ ▷ Point cloud in Oservation 1
$\mathbf{Q}_2$ ▷ Point cloud in Oservation 2
α ▷ Trade of coefficient
Output: $\mathbf{Q}_{fusion}$

if $\mathbf{Q}_1.\mathbf{s} = \mathbf{Q}_2.\mathbf{s}$ **then**
    $\mathbf{Q}_{fusion} = \mathbf{Q}_1$
else
    ▷ Probability for other unknown colors
    $\bar{p_1} = 1 - \sum(\mathbf{Q}_1.\mathbf{p})$
    $\bar{p_2} = 1 - \sum(\mathbf{Q}_2.\mathbf{p})$
    ▷ Synchronize data from $\mathbf{Q}_1$ to $\mathbf{Q}_2$
    for each *label* in $\mathbf{Q}_1.\mathbf{s}$ not in $\mathbf{Q}_2.\mathbf{s}$ **do**
        $(\mathbf{Q}_2.\mathbf{s}).push\_back(label)$
        $(\mathbf{Q}_2.\mathbf{p}).push\_back(\alpha \times \bar{p_2})$
        $\bar{p_2} = 1 - \sum(\mathbf{Q}_2.\mathbf{p})$
    end for
    ▷ Synchronize data from $\mathbf{Q}_2$ to $\mathbf{Q}_1$
    for each *label* in $\mathbf{Q}_2.\mathbf{s}$ not in $\mathbf{Q}_1.\mathbf{s}$ **do**
        $(\mathbf{Q}_1.\mathbf{s}).push\_back(label)$
        $(\mathbf{Q}_1.\mathbf{p}).push\_back(\alpha \times \bar{p_1})$
        $\bar{p_1} = 1 - \sum(\mathbf{Q}_1.\mathbf{p})$
    end for
    $\mathbf{Q}_{fusion} = \mathbf{Q}_1$
    ▷ Nomalize to probability distribution
    $\mathbf{Q}_{fusion}.\mathbf{p} = \big(\mathbf{Q}_1.\mathbf{p} \times \mathbf{Q}_2.\mathbf{p}\big) \big/ \big(\sum(\mathbf{Q}_1.\mathbf{p} \times \mathbf{Q}_2.\mathbf{p})\big)$
end if

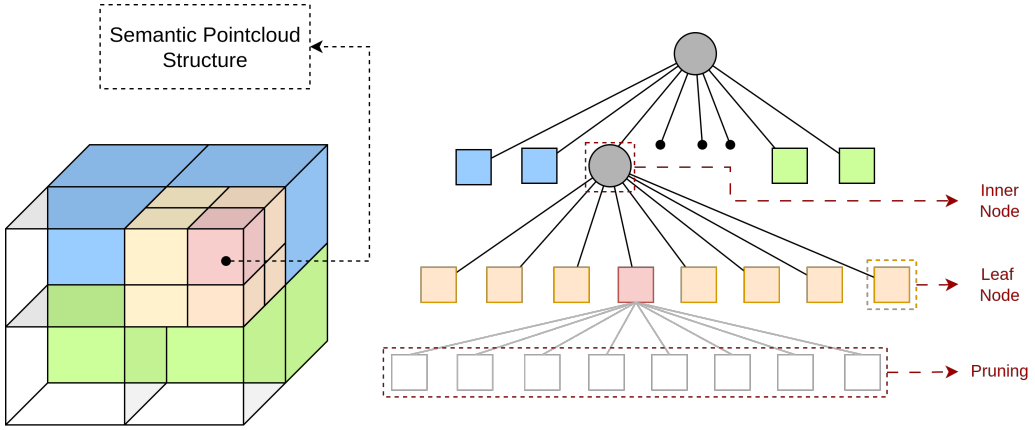


Figure 3.5: Geometric representation of Semantic Octomap (left) and Example of Octree structure (right): A gray circle signifies an inner node whose child nodes jointly span the same physical area as the inner node. A colored square indicates occupancy values sharing the same semantic, and black dots denote areas that have not been explored.

To overcome these challenges, we utilized OctoMap [61], a probabilistic 3D mapping framework that leverages an Octree data structure allowing for adaptive resolution. Compared to point cloud maps, OctoMap offers a more efficient method for storing occupancy status, thereby significantly reducing storage requirements. An Octree is a layered data structure consisting of nodes that symbolize segments of the spatial environment. Each node can have 0 or 8 child nodes, which align with the 8 subdivisions in the Euclidean 3D coordinate system [62]. Fig. 3.5 illustrates an example of a Semantic Octree data structure. Leaf nodes denote the smallest voxels, and a probabilistic model deals with issues such as noise and range measurement errors by assigning probabilities to states of occupancy or vacancy. A Semantic Octree instance stores the occupancy, color, and semantic information related to its corresponding physical space. This data structure is organized as a Semantic point cloud, further elaborated in Section 3.3.2. OctoMap is thus a preferred option for constructing maps in our system, as it addresses the drawbacks of conventional point cloud approaches. When a new 3D point is added, the log odds value for voxel i at time t ($L(i|Z_{1:t-1})$ is determined by leveraging the log odds value up to time $t-1$ ($L(i|Z_{1:t-1})$:

$$L(i|Z_{1:t}) = L(i|Z_{1:t-1}) + L(i|Z_t) \tag{3.17}$$

where,

$$L(i) = \log\left[\frac{p(i)}{1-p(i)}\right] \tag{3.18}$$

In this context, Z_t denotes the observation for a voxel at time t, while $p(i)$ represents the probability that the voxel i contains an object or obstacle. By integrating OctoMap with our semantic point cloud structure, the resulting semantic map is both memory efficient and semantically rich. This enhanced map provides a detailed understanding of the environment, enabling advanced applications such as semantic-aware navigation and task planning. The combination of probabilistic occupancy mapping with semantic fusion ensures robustness and adaptability in dynamic and unstructured environments.

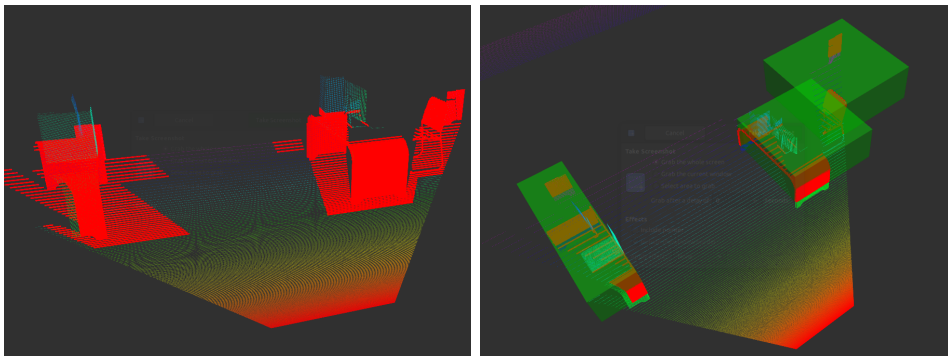
3.4 2.5D Semantic SLAM with Object Detection

2.5D Semantic SLAM with Object Detection combines the capabilities of simultaneous localization and mapping (SLAM) with object detection to create a detailed, semantically enriched representation of the environment. This approach leverages both geometric and semantic information to enhance the accuracy and utility of generated maps. The system can identify and localize objects within the environment, providing a richer context for navigation and interaction. Our main objective is to utilize the RGB-D images to build a semantic probability map from these grid maps. Firstly, RGB images are processed for object detection using a neural network model. For this purpose, we have selected the highly accurate and real-time YOLOv8 architecture. YOLOv8 is known for its fast and reliable object detection, which makes it ideal for real-time applications in dynamic environments. Subsequently, the detected objects are tracked with the BoT-SORT [63] algorithm, which ensures consistent identification over different time intervals. BoT-SORT enhances the tracking reliability by associating detected objects across frames, thereby maintaining consistent object identities over time.

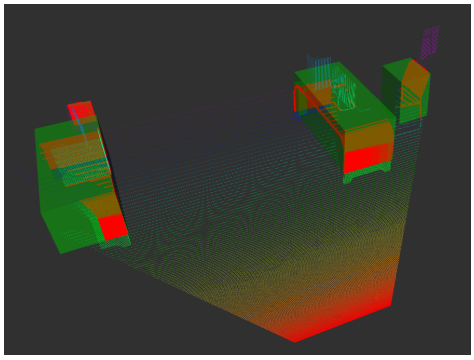
Secondly, we transform the depth images into point clouds and extract the point cloud data of the detected objects. This transformation involves converting depth information into a three-dimensional spatial representation, enabling precise localization and mapping of objects within the environment. To improve data quality, we applied a clustering method to eliminate outlier point clouds. This step is crucial for ensuring the accuracy and reliability of the point cloud data by removing noise and irrelevant points. The core process of semantic map creation involves the Cartographer method [8], which monitors the robot's pose within the environment and produces a 2D metric map using 2D scan data obtained from the point cloud. Cartographer is an

advanced SLAM algorithm known for its real-time performance and accuracy in pose estimation and map generation. Afterward, we compute object positions and project the relevant semantic information onto the robot's coordinate system. This involves integrating the detected object positions with the robot's pose to accurately place objects within the generated map. Finally, we augment the metric map by integrating semantic information and associated projection data, adopting a probabilistic approach. This probabilistic integration ensures that the semantic map reflects the uncertainties and variations in the environment, providing a robust and reliable representation.

3.4.1 Semantic knowledge understanding



(a) Raw point cloud of object class (b) Remove background step



(c) Cluster point cloud step

Figure 3.6: Steps and outcomes of semantic knowledge comprehension. The red dot represents a point cloud derived from RGB images, while the green rectangle signifies a 3D bounding box indicator.

The identification of object categories within the RGB-D camera's field-of-view (FOV) is an essential step in our approach. To achieve this, we

utilize a neural network capable of extracting semantic information on a pixel level from the images. In our study, we use the *You Only Look Once* [15] (YOLO) algorithm, recognized as the leading model for object detection and real-time performance, even on affordable embedded devices. Specifically, we have chosen Yolov8, which is the fastest, lightest, and has the highest precision-recall scores.

YOLOv8 processes the input RGB image to produce bounding boxes and probability scores for each category of identified objects. Each bounding box is defined by four parameters: the center coordinates $x(i), y(i)$, the width $w(i)$, and the height $h(i)$. YOLOv8 is highly efficient due to its backbone network, which is designed for both speed and accuracy, making it suitable for real-time applications. The network architecture includes several innovations such as CSPDarknet53 as the backbone, PANet as the path-aggregation network, and the YOLOv3 head for detection, ensuring high performance across various scenarios.

After detecting the classes of objects, we utilize BoT-SORT [63] to monitor multiple objects over different observations. BoT-SORT consists of three main components:

1. Discrete Kalman Filter: This models the object's motion in the image plane, predicting the future positions of objects based on their past states.
2. Camera Motion Compensation: This compensates for the rigid camera motion, ensuring that object tracking remains accurate even when the camera is moving.
3. IoU-Re-ID Fusion: This integrates appearance features into the tracker, combining intersection over union (IoU) with re-identification (Re-ID) metrics to maintain consistent object identities across frames.

The model objects of class i are described as follows:

$$\mathbf{x}_i = \begin{bmatrix} x(i) & y(i) & w(i) & h(i) & \dot{x(i)} & \dot{y(i)} & \dot{w(i)} & \dot{h(i)} \end{bmatrix}^T \tag{3.19}$$

$$\mathbf{z}_i = \begin{bmatrix} z_x(i) & z_y(i) & z_w(i) & z_h(i) \end{bmatrix}^T \tag{3.20}$$

Simultaneously, we obtain point clouds using the RGB-D camera. Let us denote the depth at pixel (x, y) as $D(x, y)$. Assuming the RGB-D camera follows a pin-hole model with focal lengths (f_x, f_y) and optical center (c_x, c_y), the intrinsic camera matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$, along with the extrinsic parameters $(\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{t} \in \mathbb{R}^{3\times1})$, the 3D coordinates of point k in the world coordinate

system are given by:

$$\mathbf{p}_k = \begin{bmatrix} X_k \\ Y_k \\ Z_K \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_{0\times3} & 1 \end{bmatrix} * \begin{bmatrix} D(x,y)K^{-1}x \\ D(x,y)K^{-1}y \\ D(x,y) \\ 1 \end{bmatrix} \tag{3.21}$$

In the next step, we begin by extracting complete point clouds $\mathbf{P}$ from within the object's bounding box, as determined by the object detection model. To ensure both robustness and efficiency, the point cloud $\mathbf{P}$ undergoes a pre-processing phase that includes two main steps: Euclidean-based clustering and background removal. Euclidean-based clustering is based on the concept of spatial proximity. Points that fall within a predefined Euclidean distance threshold ϵ are clustered together, as they are likely to belong to the same object or surface. The Euclidean distance $d_{euclidean}$ between any two points (x_1, y_1, z_1) and (x_2, y_2, z_2) is calculated as:

$$d_{euclidean} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.22}$$

After segmenting the point cloud into clusters, we select the largest cluster based on the number of points it contains. Fig. 3.6 illustrates the steps involved in point cloud processing.

3.4.2 Localization and mapping

To achieve robust and precise localization along with comprehensive mapping, our research uses CartoGrapher, an advanced 2D SLAM solution smoothly integrated within the Robot Operating System (ROS). This strategic combination empowers our robotic system to perform simultaneous localization and mapping, thus creating detailed representations of the surrounding environment. CartoGrapher is a powerful 2D SLAM algorithm known for its capability to produce highly accurate maps of both indoor and outdoor settings. Key aspects of CartoGrapher include:

- **Real-Time Correlative Scan Matching (RTCSM):** RTCSM aligns incoming laser scan data with the existing map to estimate the robot's pose. This method computes the transform that best matches the current scan to the submap, providing a robust initial pose estimate. This process involves correlating the scan against a probability grid, evaluating multiple possible transformations, and selecting the one with the highest matching score.

- **Pose Graph Optimization:** After initial pose estimation via RTCSM, Cartographer refines the pose estimate through pose graph optimization. This global optimization step minimizes the overall error by adjusting the poses of all nodes in the graph, ensuring a consistent and accurate map. Constraints between nodes are established based on scan-to-submap matches, loop closures, and odometry, forming a connected graph structure. Optimization algorithms, such as Ceres Solver, are employed to solve this non-linear least squares problem.

The resulting output is a two-dimensional metric map that depicts the environment, derived from the grid M and the robot's pose $\mathbf{p}_r \in \mathbb{R}^3$ within the map:

$$\mathbf{p}_r =^M \mathbf{T}_r = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{3.23}$$

where (x, y) and θ represent the position and orientation, respectively, and ${}^M\mathbf{T}_r$ is the transformation matrix between map coordinates and robot coordinates.

3.4.3 Semantic association

After extracting the point cloud of the object class, the next step is to determine the position of the object in the coordinates of the map. This process is critical to accurately integrate semantic information into the overall map. We denote the object position in the camera frame as $\mathbf{p}_o \in \mathbb{R}^{4\times4}$, calculated as the mean position of the points in the object's point cloud set $\mathbf{P}_o$:

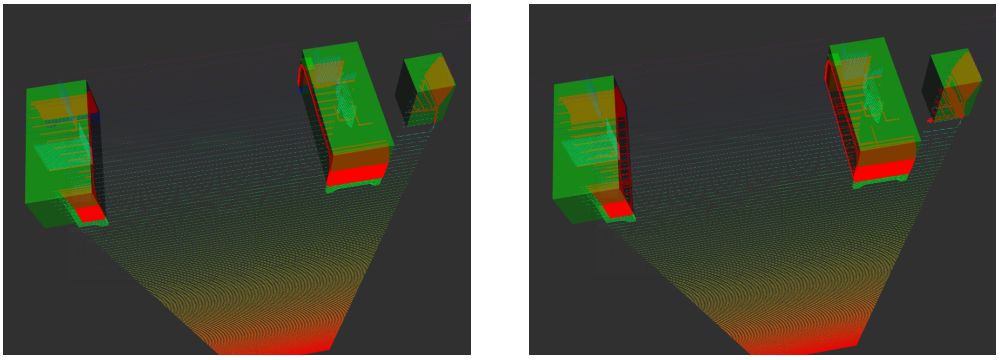
$$\mathbf{p}_o =^c \mathbf{T}_o = mean(\mathbf{P}_o) \tag{3.24}$$

To transform the position of the object from the camera frame to the map frame, we use the transformation matrix, the translation matrix, and the rotation matrix between the robot and camera coordinates, denoted as ${}^r\mathbf{t}_c$, ${}^r\mathbf{R}_c$. The position of object in the map frame $\mathbf{p}_m =^M \mathbf{T}_o \in \mathbb{R}^3$ is computed as follows:

$$\mathbf{p}_m =^M \mathbf{T}_o =^M \mathbf{T}_r +^r \mathbf{T}_c \times^c \mathbf{T}_o \tag{3.25}$$

where, ${}^r\mathbf{T}_c = \begin{bmatrix} {}^r\mathbf{R}_c & {}^r\mathbf{t}_c \\ 0_{0\times3} & 1 \end{bmatrix}$ is the transformation matrix between robot coordinates and camera coordinates

To ensure reliable navigation, it is crucial that robots maneuver gracefully around obstacles within their environment. However, traditional mapping techniques frequently cause robots to collide with objects that have hollow bases, since these are usually depicted as unoccupied grid spaces in



(a) Before projection (b) After projection

Figure 3.7: Projection of the 3D detected object point clouds

metric maps. To overcome this challenge, we utilize the RANdom-SAmple Consensus (RANSAC) [64] for the projection approach. RANSAC is an iterative method for estimating the parameters of a mathematical model from a set of observed data points that contain outliers. The fundamental idea behind RANSAC involves repetitively selecting random subsets of data points, proposing models, and assessing how well these models agree with the data.The steps in RANSAC include:

1. Random Sampling: Randomly select a subset of the original data points. This subset should be as small as possible while allowing the model parameters to be computed.
2. Model Fitting: Fit the model to the selected subset of data points.
3. Consensus Evaluation: Determine how many of the data points fit the model within a predefined tolerance. These points are considered inliers.
4. Iteration: Repeat the above steps for a fixed number of iterations or until a model with a satisfactory consensus is found.
5. Model Refinement: Once the best model is found, refine the model parameters using all inliers.

For example, to represent a plane $\mathbb{P}$ that could correspond to a "chair seat" or a "table top," RANSAC helps to identify this plane amidst noise and outliers. The plane can be defined by the equation:

$$\mathbb{P} : \mathbf{n}^T \mathbf{p}_m + D = 0 \tag{3.26}$$

where $\mathbf{n} \in \mathbb{R}^3$ is the normal vector to the plane, and D is the distance from the origin to the plane along the normal vector, determining the plane's offset.

Once the plane $\mathbb{P}$ has been segmented using RANSAC, each point $\mathbf{p}_k$ on this plane is expressed in the camera frame with the z axis oriented upward. Therefore, the projection onto the map plane can be achieved by aligning the z coordinate of the detected plane with that of the map plane. This projection process ensures that the object is correctly represented in the map, improving navigation and obstacle avoidance. Fig.
reffig:project illustrates the projection process.

3.4.4 Probabilistic map representation

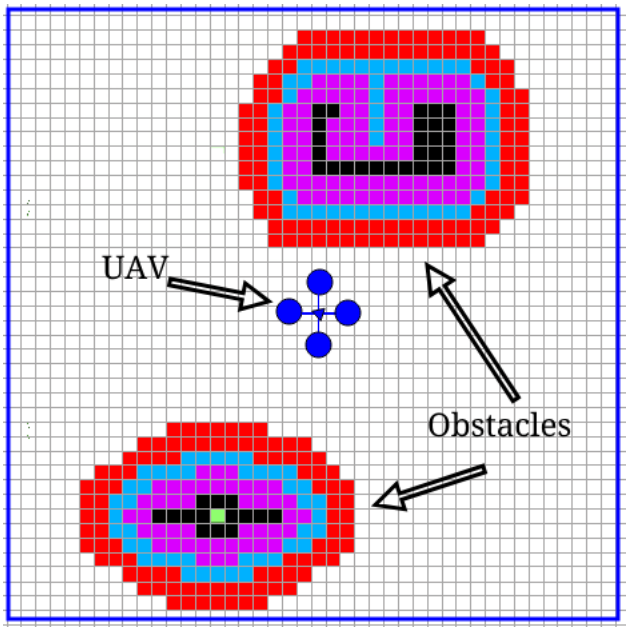


Figure 3.8: Probabilistic map representation

To improve scene understanding and decision-making in real-world settings, we present a probabilistic semantic representation framework that effortlessly integrates with the 2D costmap. The 2D costmap provides a basic grid-based depiction of the environment, incorporating crucial navigation details like obstacle positions and movement costs. Fig. 3.8 illustrates a grid costmap where each cell is assigned a value from 0 to 255, indicating the probability—ranging from 0% to 100%—of the cell being occupied. Expanding on this, our system adds a layer of semantic depth to the costmap by associating each grid cell with probabilistic semantic properties. Rather than just marking cells as obstacles or free space, we assign probabilities to indicate the likelihood of various object classes within each cell. This

flexible representation allows our autonomous system to not only recognize the spatial distribution of objects but also assess the uncertainty tied to each detection. By combining costmap data with probabilistic semantics, our system can make well-informed navigation decisions that consider the probability of encountering specific objects, thereby improving safety and flexibility in complex, dynamic environments. This method enhances the traditional 2D costmap with semantic insights, opening up new possibilities for dependable and context-aware navigation.

In summary, the integration of probabilistic semantics with the 2D costmap transforms it from a simple grid-based representation into a sophisticated tool that reflects both spatial and semantic attributes. This approach unlocks new capabilities for autonomous systems, improving their ability to navigate and interact intelligently in various real-world scenarios.

Chapter 4

Experimental Results

Within chapter, we outline the experimental setup used to evaluate the performance of the proposed Semantic SLAM system. We describe the datasets employed, including both real-world and simulated environments, and the evaluation metrics used to assess system performance. The chapter presents an in-depth examination of the results obtained from various experiments, comparing our method with existing state-of-the-art approaches. We explore the precision, computational efficiency, and robustness of the system in different scenarios. Based on empirical data, we offer insight into the strengths and limitations of the proposed system.

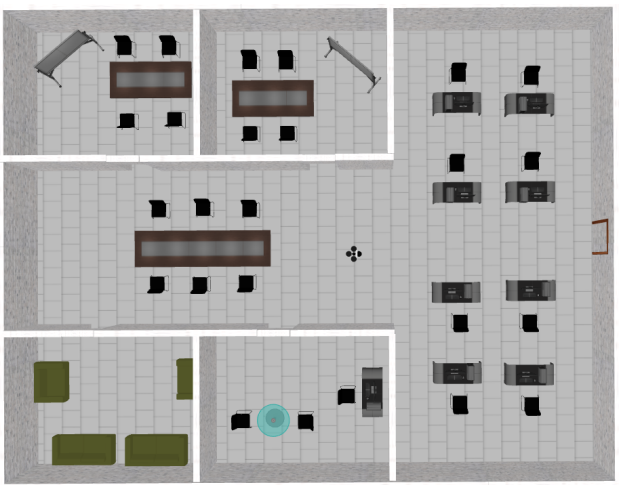
4.1 Experimental Setup

Our experimental setup is designed to assess the efficacy of our proposed Semantic SLAM solution, with a focus on both semantic mapping and pose estimation accuracy. We utilize the NVIDIA Jetson Xavier AGX as the primary processing unit, chosen for its high computational power and energy efficiency, which are crucial for real-time processing on UAV platforms.

4.1.1 UAVs and Gazebo Simulation

As shown in Fig. 4.1, we conducted the experimental tests of our proposed system utilizing the Hummingbird UAV platform, which features a RealSense D455 camera. The Hummingbird UAV is noted for its light construction, enabling swift flight maneuvers and precise navigation in intricate and changing scenarios, like those seen in challenging search and rescue operations. It features advanced flight control algorithms, which guarantee steady and regulated flight performance throughout the experiments. The RealSense D455 camera improves the UAV's functionality by providing RGB-D data. It has a horizontal field of view (FOV) of 90 degrees, a vertical FOV of 58 degrees, a depth FOV of 98 degrees, an image resolution of 640×480, and a frame rate of 60 Hz.

(a) Home environment



(b) Office environment with hollow objects

Figure 4.1: UAV and Gazebo environment simulation

Our experiments were conducted in two distinct environments to test the versatility and robustness of the system. The first environment (Fig.4.1a) simulates an office setting with various rooms that contain tables, chairs, and other office furniture. This setup is designed to mimic an indoor search and rescue scenario where the UAV must navigate through narrow spaces and identify objects such as desks and chairs. The second environment (Fig.4.1b) represents a home setting with a bedroom, living room, kitchen, and dining area. This environment tests the UAV's ability to navigate and map a more residential and varied setting, ensuring the system's effectiveness in different real-world applications. Both environments were modeled in Gazebo, a versatile simulation tool that provides realistic physics and sensor

data, allowing for comprehensive testing of UAV systems in various scenarios. The UAV's path was planned and executed to cover all accessible areas within these environments, and its performance was evaluated based on navigation accuracy, object detection, and mapping fidelity.

4.1.2 Datasets

4.1.2.1 Real Publicly Available Dataset

We employed the TUM publicly available RGB-D dataset [65] for benchmarking our system. The TUM dataset provides high-quality, synchronized RGB-D data collected from various indoor environments. This dataset is widely used for evaluating SLAM algorithms due to its extensive and diverse scenarios, which include office environments, living rooms, and more. The TUM dataset's comprehensive annotations and ground truth data enable us to perform rigorous quantitative evaluations of our system's effectiveness regarding precision and stability in practical scenarios. The trajectory concludes where it started, creating a significant loop closure. Several important aspects such as:

- Duration: $87.09s$
- Duration with ground-truth: $87.10s$
- Ground-truth trajectory length: $21.455m$
- Avg. translational velocity: $0.249m/s$
- Avg. angular velocity: $10.188deg/s$
- Trajectory dimension: $5.12m \times 4.89m \times 0.54m$

4.1.2.2 Object Segmentation Dataset

For object segmentation tasks, we leveraged the SUNRGBD dataset [66], a large-scale dataset containing RGB-D images from indoor scenes. The SUN-RGBD dataset is specifically designed for semantic segmentation, providing pixel-wise annotations for a wide range of object classes commonly found in indoor environments. This dataset allows us to train and validate our segmentation algorithms, ensuring that our system can accurately identify and segment objects in various complex settings. The dataset includes a total of $10,335$ images spanning 38 semantic categories, with $5,285$ images dedicated to training and $5,050$ reserved for validation.

4.1.2.3 Object Detection Dataset

In addition to using publicly available datasets, we collected and annotated our own dataset for object detection tasks. This custom dataset was created by capturing RGB-D images in various indoor environments and manually labeling the objects present in the scenes. Within the scope of this thesis, the training images encompass 6 categories of objects: chair, coffee table, conference table, sofa, whiteboard, and desk. The chair, desk, coffee table, and whiteboard are all characterized by fully hollow bottoms; the conference table features a partially hollow bottom, and the sofa displays a non-hollow bottom. To enhance the diversity and robustness of our object detection models, we augmented the dataset using RoboFlow, a powerful tool for data augmentation and dataset management. The augmentation process included transformations such as rotation, scaling, and flipping, which help to improve the generalization capability of our detection models. Thus, we trained the YOLOv8 object detector using over 500+ distinct images.

4.1.3 Evaluation Metrics

To rigorously assess the performance of our semantic SLAM system, we employ a combination of quantitative and qualitative evaluation metrics. These metrics provide a comprehensive understanding of the system's accuracy, robustness, and overall effectiveness in various scenarios.

1. Quantitative Metrics: The error between the aligned estimation $\hat{\mathbf{X}}'$ and the ground truth $\mathbf{X}^{gt}$ can be expressed as:

$$\Delta\mathbf{X}_i = \{\Delta\mathbf{R}_i, \Delta\mathbf{p}_i, \Delta\mathbf{v}_i\} \tag{4.1}$$

satisfies:

$$\begin{aligned} \mathbf{R}_i &= \Delta\mathbf{R}_i\hat{\mathbf{R}}'_i \\ \mathbf{p}_i &= \Delta\mathbf{R}_i\hat{\mathbf{p}}'_i + \Delta\mathbf{p}_i \\ \mathbf{v}_i &= \Delta\mathbf{R}_i\hat{\mathbf{v}}'_i + \Delta\mathbf{v}_i \end{aligned} \tag{4.2}$$

the error $\Delta\mathbf{x}_i$ can be calculated by:

$$\begin{aligned} \Delta\mathbf{R}_i &= \mathbf{R}_i(\hat{\mathbf{R}}'_i)^T \\ \Delta\mathbf{p}_i &= \mathbf{p}_i - \Delta\mathbf{R}_i\hat{\mathbf{p}}'_i \\ \Delta\mathbf{v}_i &= \mathbf{v}_i - \Delta\mathbf{R}_i\hat{\mathbf{v}}'_i \end{aligned} \tag{4.3}$$

   - Root Mean Square Error (RMSE) of Absolute Trajectory Error (ATE): The ATE measures the deviation between the predicted

trajectory and the ground truth trajectory. The RMSE of ATE is determined using the following formula:

$$\begin{aligned} ATE_{rot} &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}||\angle\Delta\mathbf{R}_i||^2} \\ ATE_{pos} &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}||\Delta\mathbf{p}_i||^2} \end{aligned} \tag{4.4}$$

where $\angle(\cdot)$ signifies converting the rotation matrix into an angle-axis representation and using the rotation angle as the error metric.

- Relative Pose Error (RPE): The RPE evaluates the local accuracy of the estimated trajectory by comparing the relative motion between K pairs of poses from $\hat{\mathbf{X}}$. The deviation $\delta\mathbf{d}_k$ for the pair of states $\mathbf{d}_k = \{\hat{\mathbf{x}}_s, \hat{\mathbf{x}}_e\}$ is calculated using the initial state $\hat{\mathbf{x}}_s$ and the adjusted second state $\hat{\mathbf{x}}_e$ is:

$$\begin{aligned} \delta\boldsymbol{\phi}_k &= \angle\delta\mathbf{R}_k = \angle\mathbf{R}_e(\hat{\mathbf{R}}'_e)^T \\ \delta\mathbf{p}_k &= ||\mathbf{p}_e - \delta\mathbf{R}_k\hat{\mathbf{p}}'_e||_2 \end{aligned} \tag{4.5}$$

RPE is computed as follows:

$$\begin{aligned} RE_{rot} &= \{\delta\boldsymbol{\phi}_k\}_{k=0}^{K-1} \\ RE_{pos} &= \{\delta\mathbf{p}_k\}_{k=0}^{K-1} \end{aligned} \tag{4.6}$$

- Mean Average Precision (mAP): The mAP is utilized to assess the effectiveness of object detection, determined by taking the average of precision values across various recall levels:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP(i) \tag{4.7}$$

where $AP(I)$ represents the mean precision for the i-th object class, where N denotes the total count of object classes.

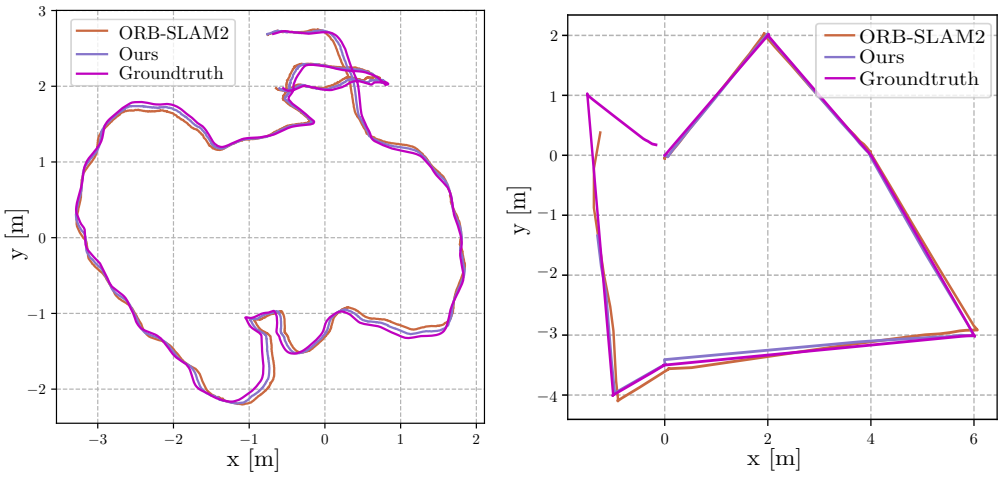
- Precision and Recall: Precision and recall are essential measures for evaluating the effectiveness of object detection and segmentation. Precision is defined as the proportion of correctly identified positive detections to all identified positive detections, whereas recall is the proportion of correctly identified positive detections to all actual positive instances ground truth positives:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN} \tag{4.8}$$

where TP represents the count of true positives, FP represents the count of false positives, and FN represents the count of false negatives.

2. Qualitative Metric: Qualitative metrics involve visual inspection and analysis of the system's output. This includes evaluating the quality of the produced maps, the precision of object segmentation, point cloud projection, and the overall navigation behavior of the UAV in different environments.

4.2 Experimental Resuls

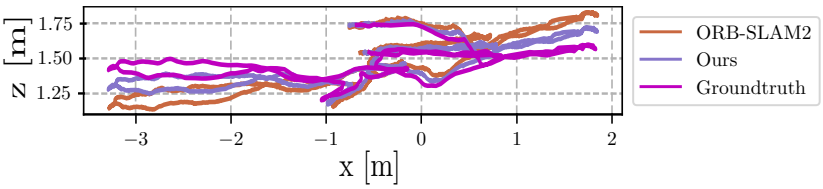


(a) TUM Dataset (b) Gazebo Dataset

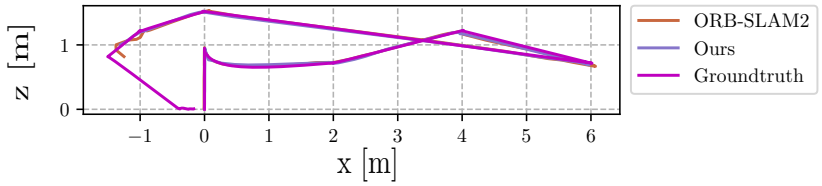
Figure 4.2: The comparison of trajectory for ORB-SLAM2, Our system and ground truth in X-Y axis

4.2.1 Pose Estimation Results

Figs. 4.2 and 4.3 show the trajectory comparison for the ORB-SLAM2, our proposed system, and the ground truth in the X-Y and X-Z axes respectively for the TUM and Gazebo datasets. In the trajectory comparison within the X-Y axis, our method demonstrated a mean deviation from the ground truth of $0.15(m)$ for the TUM data set and $0.12(m)$ for the Gazebo data set, compared to ORB-SLAM2 deviations of $0.20(m)$ and 0.18 (m), respectively. Similarly, on the X-Z axis, our method showed mean deviations of $0.10(m)$



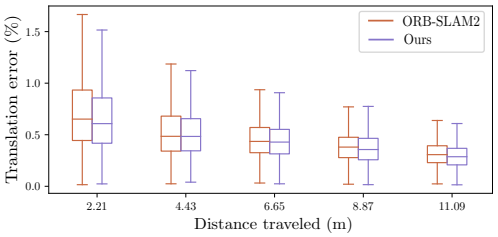
(a) TUM Dataset



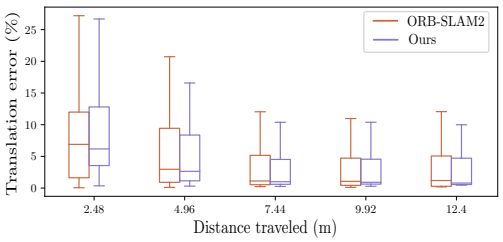
(b) Gazebo Dataset

Figure 4.3: The comparison of trajectory for ORB-SLAM2, our sytem and ground truth in X-Z axis

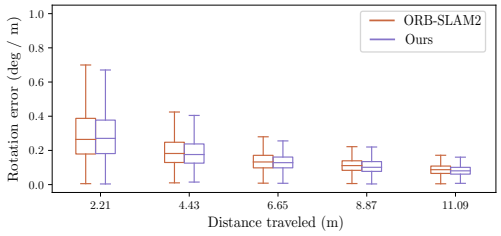
(TUM) and $0.08(m)$ (Gazebo), while ORB-SLAM2 recorded $0.15(m)$ and $0.14(m)$. The proposed system tends to show better adherence to the ground truth, especially in areas where the path changes direction sharply.



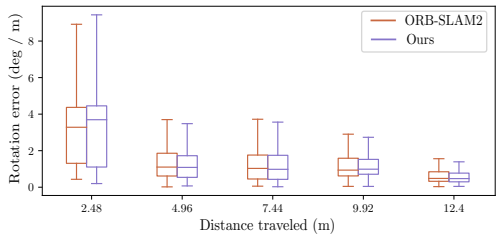
(a) Translation Error in TUM dataset



(b) Translation Error in Gazebo dataset



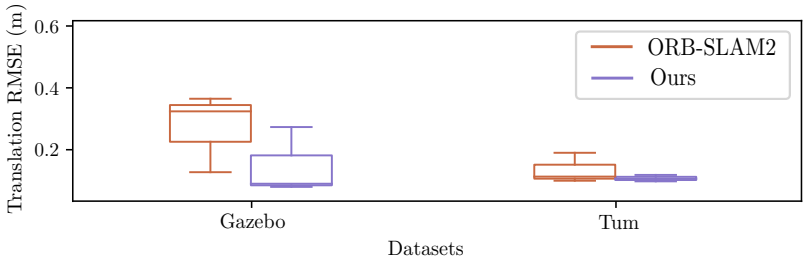
(c) Rotation Error in TUM dataset



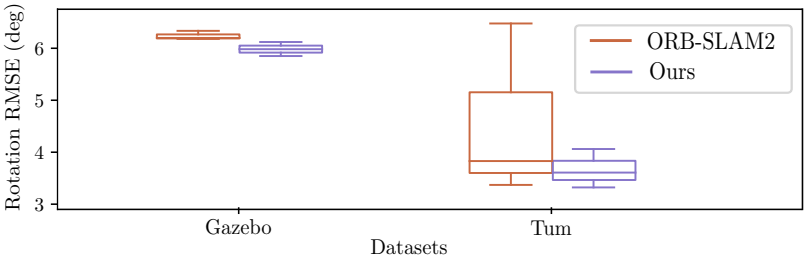
(d) Rotation Error in Gazebo dataset

Figure 4.4: Comparison of Relative Rose Error (RPE) between ORB-SLAM2 and Our system

For translation error (RPE), Fig. 4.4 illustrates that our system and ORB-



(a) Translation Error



(b) Rotation Error

Figure 4.5: The comparison of ORB-SLAM2 and Our system based on the RMSE of ATE

SLAM2 exhibited accurate pose estimation and smooth motion throughout the environments. Our method maintained lower error percentages across all distances. Specifically, at a distance of 11.09(m) in the TUM dataset, our method had a 0.27% error compared to ORB-SLAM2's 0.32%. This trend of lower translation errors for the proposed system is consistent across all measured distances, indicating its superior accuracy. The translation errors in the Gazebo dataset are generally higher than those in the TUM dataset. Despite this, the proposed system still exhibits lower errors compared to ORB-SLAM2 across all distances. For example, at 12.4(m) in the Gazebo dataset, our method exhibited a 2.17% error against ORB-SLAM2's 2.22%. This suggests that the proposed system maintains a better accuracy even in more challenging conditions. The rotation error analysis in the TUM dataset highlights that the proposed system has lower errors compared to ORB-SLAM2. At a distance of 11.09(m) ORB-SLAM2 has a rotation error of approximately 0.09(deg/m), whereas the proposed system's error is about 0.08(deg/m). Similarly to translation errors, rotation errors are higher in the Gazebo dataset. However, the proposed system consistently shows lower rotation errors. For instance, at a distance of 12.4(m), the rotation error for ORB-SLAM2 is around 0.69(deg/m), while for the proposed system, it

is approximately $0.65(deg/m)$. This further confirms the robustness and accuracy of the proposed system.

In addition, Fig. 4.5 illustrates the RMSE of ATE for all frames across both frameworks, confirming that our system maintains consistent pose estimation performance. For translation error, our method achieved $0.10(m)$ in the Gazebo dataset and $0.09(m)$ in the TUM dataset, compared to ORB-SLAM2 $0.33(m)$ and $0.11(m)$. In rotation error, our method recorded $5.93(deg)$ (Gazebo) and $3.50(deg)$ (TUM), while ORB-SLAM2 had $6.15(deg)$ and $3.86(deg)$. These results collectively affirm that our proposed method delivers improved precision and robustness in pose estimation over ORB-SLAM2, as evidenced by lower deviations from ground truth, reduced translation and rotation errors, and lower RMSE of ATE across different datasets.

4.2.2 Semantic Extraction Results

The training process of various segmentation networks is depicted in 4.6, showcasing the performance of six different networks: PSPNet [26], ICNet [25], SegNet [21], UNet [22], FRRNs [24], and FCNs [20], each trained over 100 epochs using a batch size of 2 on a Nvidia T4. Among these, PSPNet demonstrates the highest performance, achieving an accuracy of approximately 0.78 by the end of the training period. This superior performance highlights PSPNet's effectiveness in capturing and segmenting semantic information, making it a reliable choice for our application. ICNet follows with an accuracy of around 0.70, achieving a balance between accuracy and computational efficiency, essential for real-time tasks. The optimization process employed standard stochastic gradient descent, with parameters including a weight decay of $1e-3$, a momentum value of 0.9, and a learning rate set at 0.01. The best-performing model was chosen based on these training parameters.

4.2.2.1 Semantic Segmentation

SegNet and Unet show steady improvements throughout the training, reaching accuracies of about 0.66 and 0.68, respectively. These networks, with their encoder-decoder architectures and symmetric designs, are well suited for segmentation tasks and show potential for further improvement with additional fine-tuning. The FRRNs (A & B) exhibit moderate performance, with accuracy of approximately 0.65 and 0.63, respectively. While they handle high-resolution images efficiently, their complex architectures might benefit from more extensive training. The FNC models, FCN-16s and FCN-8s,

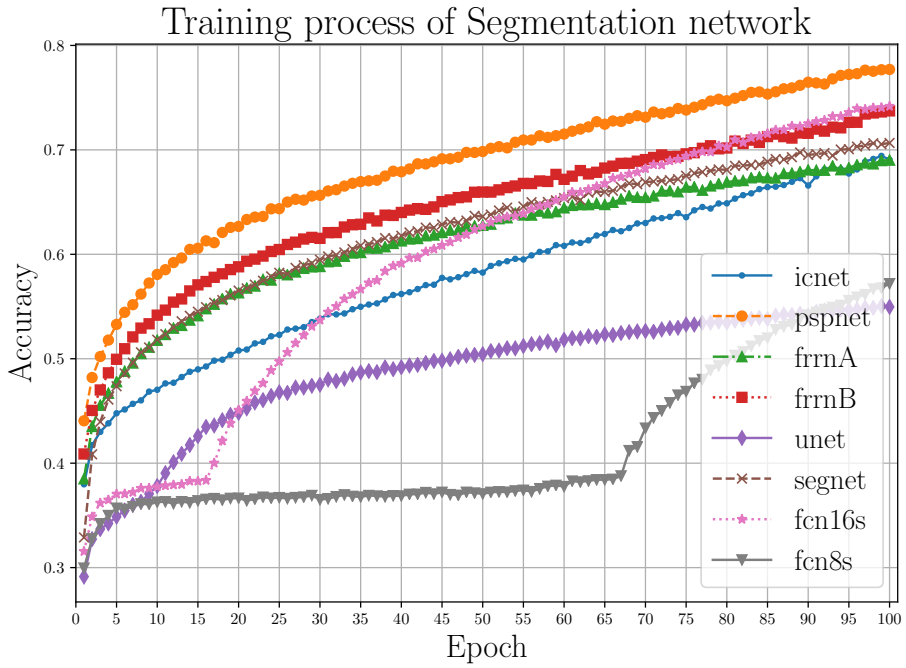


Figure 4.6: Training Models Assessment

achieve lower accuracies of around 0.50 and 0.55, respectively, indicating that while foundational, they are outperformed by newer architectures like PSPNet and ICNet. Overall, these results validate PSPNet as the leading network for our semantic segmentation tasks, with ICNet as a viable alternative, while SegNet and Unet also show promise for further development. Among the models, PSPNet exhibited superior accuracy performance, prompting its selection as the segmentation model for integration into our system.

4.2.2.2 Object Detection

Table 4.1 provides a comprehensive comparison of the performance metrics of various detection models, including our model. In particular, we evaluate our methodology with Yolov3, which was used in a previous work by D. Bersan [55]. Our model achieves the highest mAP of 98.2%, outperforming all other models in terms of detection accuracy. This indicates that our model is highly effective in identifying and localizing objects within the environment. Furthermore, our model maintains a high precision of 92.0% and a recall of 92.9%, demonstrating its robustness and reliability in detecting objects correctly.

Among the models compared, RTMDet and Fast R-CNN also perform

Table 4.1: Results of detection models

Models	**mAP (IOU=0.5)**	**Parameter**	**Precision**	**Recall**
Yolov3 [55]	90.0%	8.7M	85.9%	84.6%
Fast R-CNN	95.2%	12.9M	90.2%	92.0%
MobileNet	88.6%	4.6M	91.1%	83.7%
Yolov4	94.8%	60.0M	82.6%	86.4%
RTMDet	95.9%	52.3M	91.5%	88.4%
Yolov5	94.0%	7.0M	87.0%	92.7%
Yolov7	94.8%	**3.7**M	83.8%	96.2%
Our	**98.2**%	11.1M	92.0%	92.9%

well, with mAP values of 95.9% and 95.2%, respectively. However, RTMDet has a significantly higher number of parameters ($52.3M$) compared to our model, making it less efficient in terms of computational resources. Fast R-CNN, while having a higher mAP than most other models, also comes with a relatively high parameter count of $12.9M$. Yolov7 achieves a notable balance with a mAP of 94.8%, precision of 83.8%, and the lowest parameter count at $3.7M$. This model demonstrates excellent efficiency and could be an alternative for applications where computational resources are limited. Overall, the results highlight the superiority of our model in terms of detection accuracy, with an optimal balance of precision and recall, making it well-suited for real-time applications requiring accurate and reliable object detection.

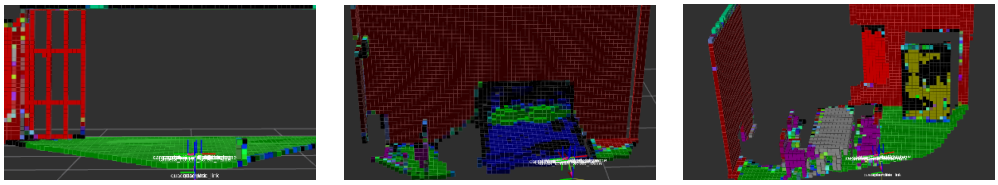
4.2.3 3D Semantic Mapping Results

Figure 4.7 provides a detailed visual representation of the semantic mapping process. Subfigure 4.7a displays the input images captured from the camera, which serve as the starting point for our mapping system. Subfigure 4.7b shows the results of the semantic segmentation process, where each object in the input images is labeled with a specific color corresponding to its class. This step is crucial to distinguish between different objects in the environment. Subfigure 4.7c illustrates the color point cloud generated from the input images. This color point cloud incorporates both the geometric information and the RGB data, providing a rich and detailed representation of the environment. Subfigure 4.7d demonstrates the 3D semantic mapping, where the segmented objects are accurately placed in the 3D space, creating a semantically enriched map. Finally, Subfigure 4.7e presents the overall 3D

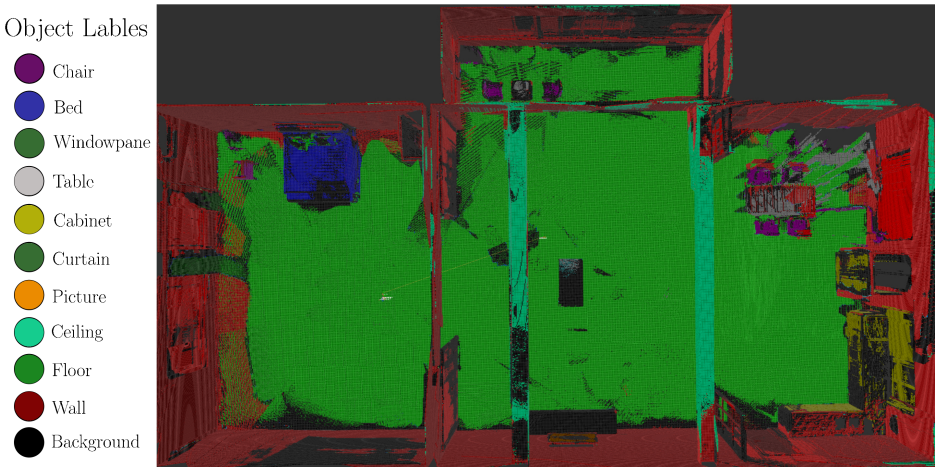
(a) Input image from camera

(b) Semantic segmentation from input images

(c) Color point cloud from input images



(d) 3D semantic mapping of input images



(e) Overall 3D semantic mapping

Figure 4.7: 3D visual representation of the obtained semantic maps

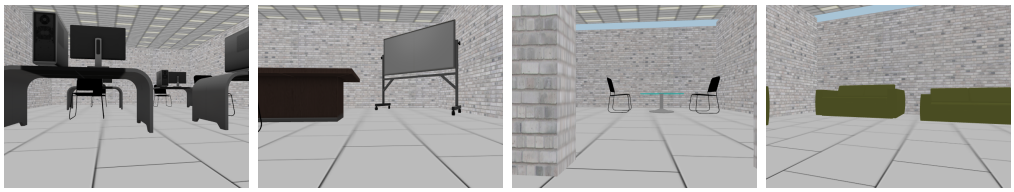
semantic map of the environment. This map integrates all individual 3D semantic mappings into a cohesive and comprehensive representation. The legend in the figure indicates the object labels used in the mapping process, such as chairs, beds, tables, and walls, each represented by a distinct color.

The results showcase the effectiveness of our semantic SLAM system in generating detailed and accurate 3D semantic maps. The system successfully identifies and localizes various objects within the environment, enhancing the map's utility for navigation and interaction. By incorporating semantic information into the SLAM process, our system provides a richer context for understanding the environment, making it highly suitable for applications such as autonomous navigation, search and rescue, and indoor mapping. In addition, the implementation of the proposed system on the Jetson Xavier AGX platform, operating at $2Hz$, where the object segmentation phase takes $40ms$ per frame. These mapping results highlight the system's ability to achieve real-time semantic mapping performance.

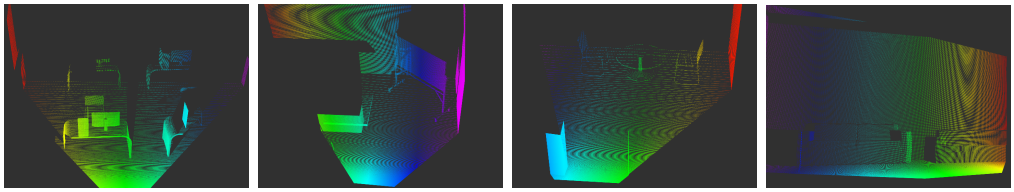
4.2.4 2.5D Semantic Mapping Results

Fig. 4.8 provides the experimental results for the visualization of the generated semantic maps, where red points signify the point clouds of identified objects, and blue text denotes the labels of these objects. Subfigure 4.8a shows the input images captured from the camera, which serve as the initial data for our mapping system. Subfigure 4.8b displays the point clouds generated from these input images, incorporating both geometric information and RGB data to create a detailed representation of the environment. Subfigure 4.8c illustrates the object detection results from the input images. The detected objects are highlighted and segmented, facilitating their identification and localization within the point cloud data. Subfigure 4.8d shows the clustering of point clouds based on the detected objects, enhancing the distinction between different objects and surfaces. Subfigure 4.8e demonstrates the projection of point clouds onto a 2D plane, representing the objects' positions and dimensions in a more manageable format. This step is crucial for integrating the semantic information into the SLAM process, enabling accurate and efficient mapping. Finally, Subfigure 4.8f presents the probabilistic map representation of the environment. This map integrates the semantic information and the projected point clouds, creating a comprehensive and enriched 2.5D representation. The probabilistic approach allows for handling uncertainties in object detection and localization, providing a more robust map for navigation and interaction.

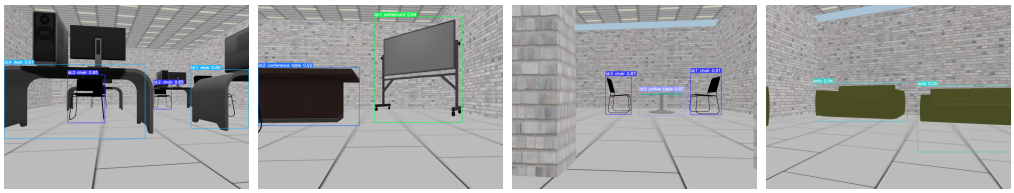
Additionally, minor delays in the point cloud processing stage sometimes resulted in objects being mapped to incorrect positions in the 2D map, espe-



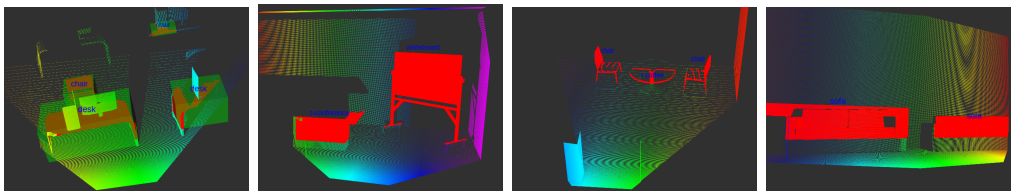
(a) Input image from camera



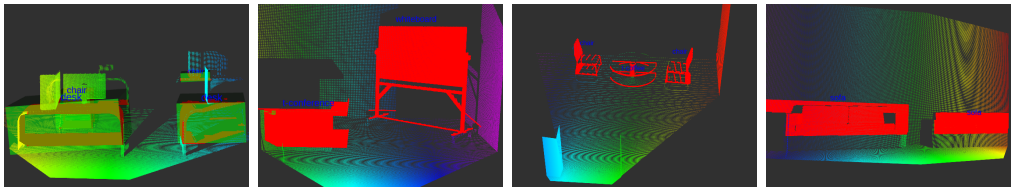
(b) Point cloud from input images



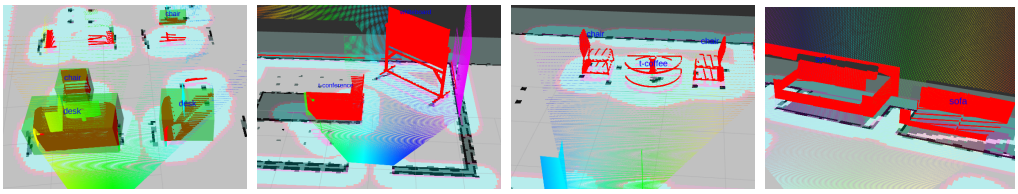
(c) Object detection from input images



(d) Point cloud clustering

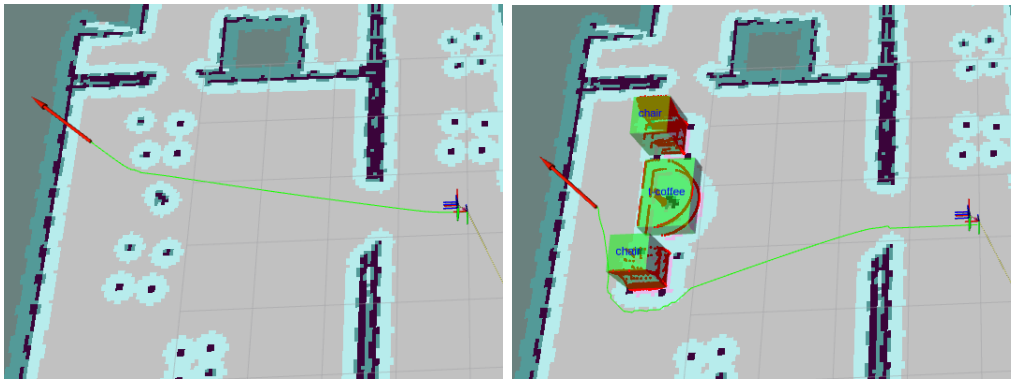


(e) Point cloud projection



(f) Probabilistic map representation

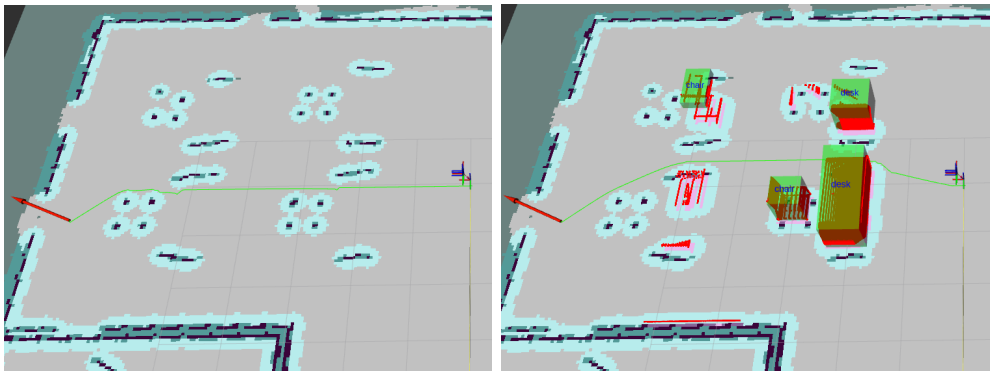
Figure 4.8: Visual representation of the obtained semantic maps



(a) Obstacle avoidance path using only metric map

(b) Obstacle avoidance path using only our map

Figure 4.9: Avoiding coffee table obstacles



(a) Obstacle avoidance path using only metric map

(b) Obstacle avoidance path using only our map

Figure 4.10: Avoiding chair and desk obstacles

cially during the robot's turns. To mitigate this problem, we synchronized the object detection and point cloud processing stages. Moreover, experimental results indicate that the projection stage can accurately determine the object's pose, while the probabilistic map representation stage provides enhanced cell-based map information compared to the binary values of 0 for free cells and 1 for occupied cells in the standard metric map.

4.2.5 Safety Navigation

To demonstrate the capability of safe navigation, we evaluated our approach using chair obstacles, desk obstacles, and coffee table obstacles, which are en-

tirely hollow at the bottom.. Fig. 4.9 and Fig. 4.10 illustrate the effectiveness of our semantic mapping system in enhancing obstacle avoidance compared to traditional metric maps. Subfigure 4.9a shows the obstacle avoidance path using only a traditional metric map. The path planning algorithm is limited by the lack of semantic information, leading to potential navigation issues around objects like coffee tables, which are often misrepresented in metric maps. In contrast, Subfigure 4.9b demonstrates the obstacle avoidance path using our semantic map. The incorporation of semantic information allows the system to accurately recognize and navigate around the coffee table, resulting in a safer and more efficient path. Similarly, Subfigure 4.10a presents the obstacle avoidance path using a traditional metric map when navigating around chairs and desks. The lack of detailed object information in the metric map can lead to suboptimal paths and potential collisions. Subfigure 4.10b shows the improved obstacle avoidance path using our semantic map. The system's ability to accurately detect and understand the positions and dimensions of chairs and desks results in a more precise and safe navigation path.

The results clearly indicate that our semantic mapping system significantly enhances the UAV's ability to navigate complex environments safely. By incorporating semantic information into the navigation process, the system can avoid obstacles more effectively than with traditional metric maps alone. This capability is crucial for applications such as autonomous navigation, search and rescue, and indoor mapping, where safety and efficiency are paramount. Moreover, the object detection stage takes only 0.25 seconds, and the point cloud processing stage takes 0.5 seconds. This real-time efficiency highlights the practical usability of our method, making it highly suitable for time-critical applications.

Chapter 5

Conclusions and Future Work

This chapter summarizes the research performed in this thesis, along with a concise discussion of certain limitations and possible future directions.

5.1 Conclusions

In this thesis, we have presented a comprehensive approach to enhancing the capabilities of UAVs in real-time semantic SLAM and safe navigation within complex environments. Our proposed system integrates advanced techniques in visual SLAM, semantic segmentation, and object detection to construct detailed and semantically enriched 3D and 2.5D maps. Implemented on the Jetson Xavier AGX platform, our system demonstrates real-time processing capabilities with high accuracy and efficiency.

We have demonstrated the effectiveness of our system through extensive evaluations using both simulated and real-world datasets. The experimental results indicate that our semantic mapping system significantly improves the UAV's ability to perceive and navigate its surroundings. The enhanced maps, enriched with semantic information, provide a richer context for navigation and interaction, leading to safer and more efficient path planning.

Our system's ability to accurately detect and localize objects within the environment, coupled with the robust semantic association and probabilistic mapping techniques, underscores its potential for a wide range of applications, including autonomous navigation, search and rescue, and indoor mapping. The integration of semantic information into the SLAM process represents a significant advancement in the field of autonomous robotics.

5.2 Limitations

Despite the promising results, several limitations exist in our current approach:

- **Dependency on RGB-D Input Quality**: Poor lighting conditions, reflections, and occlusions can negatively impact the accuracy of object detection and segmentation.
- **Computational Requirements**: Real-time semantic processing, while manageable on the Jetson Xavier AGX, may pose challenges for smaller, resource-constrained UAV platforms.
- **Cumulative Errors in SLAM**: Potential for cumulative errors in the SLAM process can affect the overall accuracy of the generated maps.
- **Focus on Indoor Environments**: The current approach primarily targets indoor environments, presenting challenges when extending to outdoor scenarios.

5.3 Future Research Directions

Future research can build upon the foundations laid in this thesis by exploring several promising directions.

- **Enhancement of Lightweight Extraction:**
  - Focus on developing lightweight models for semantic extraction to ensure real-time processing capabilities on a wider range of UAV platforms, including smaller and more resource-constrained systems.
  - Leverage advancements in model compression and efficient neural network architectures to maintain high accuracy while reducing computational overhead.
- **Robust Data Association:**
  - Enhance probabilistic data association techniques to improve the robustness and accuracy of object detection and mapping in dynamic and cluttered environments.
  - Develop advanced algorithms for probabilistic data association that can handle multiple hypotheses and uncertainties in object tracking and localization.
- **Active Semantic Perception SLAM:**
  - Incorporate active SLAM techniques to enable the UAV to make informed decisions about its movements, optimizing the exploration and mapping process.
  - Develop strategies for the UAV to actively seek out areas of uncertainty or interest, enhancing the overall map quality and completeness.

- **Efficient Map Representation:** Leveraging hierarchical structural method to enhance computational efficiency.
- **Long-term Operation in Dynamically Changing Environments:**
  - Implement continual SLAM methods that allow the system to continuously learn and update the map over extended periods, accommodating changes in the environment.
  - Focus on incremental learning techniques that enable the SLAM system to adapt to new observations without retraining from scratch.

In summary, while this thesis presents a significant advancement in real-time semantic SLAM and safe navigation for UAVs, there are numerous opportunities for future research to further enhance and expand the capabilities of the proposed system. By focusing on lightweight extraction, active semantic SLAM, and probabilistic data association, future work can address the current limitations and push the boundaries of autonomous navigation and mapping.

Publications and Awards

The master studies resulted in several publications which are detailed in this section. The entries are organized by the first author's name and year of publication.

Publications related to this Thesis

[1] Thanh Nguyen Canh, Van-Truong Nguyen, Xiem HoangVan, Armagan Elibol, and Nak Young Chong, "S3m: Semantic Segmentation Sparse Mapping for UAVs with RGB-D Camera," *2024 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2024, pp. 899–905. http://doi.org/10.1109/SII58957.2024.10417379.

[2] Thanh Nguyen Canh, Armagan Elibol, Nak Young Chong, and Xiem HoangVan, "Object-oriented Semantic Mapping for Reliable UAVs Navigation," *2023 12th International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, 2023, pp. 139–144. https://doi.org/10.1109/ICCAIS59597.2023.10382351.

Other Publications

Journal

[3] Thanh Nguyen Canh, Son Tran Duc, Huong Nguyen The, Trang Huyen Dao, and Xiem HoangVan, "Optimal Design and Fabrication of Frame Structure for Dual-Arm Service Robots: An Effective Approach for Human-Robot Interaction," *Engineering Science and Technology, an International Journal*, 2024. (**IF: 5.1**). https://doi.org/10.1016/j.jestch.2024.101763.

[4] Thanh Nguyen Canh, Anh Tuan Pham, and Xiem HoangVan, "Design of Deep Reinforcement Learning Approach for Traffic Signal Control

at Three-way Crossroads," *Public Transport*, 2024. (**IF: 2.3, Submitted, Under Review**).

[5] Thanh Nguyen Canh, Du Ngoc Trinh, and Xiem HoangVan, "M-Calib: A Monocular 3D Object Localization using 2D Estimates for Industrial Robot Vision System," *Journal of Automation, Mobile Robotics and Intelligent Systems*, 2024. (**IF: 0.55, Accepted**).

International Conference

[6] Thanh Nguyen Canh, Xiem HoangVan, and Nak Young Chong, "Enhancing Social Robot Navigation with Integrated Motion Prediction and Trajectory Planning in Dynamic Human Environments," *The 24th International Conference on Control, Automation and Systems (ICCAS)*, 2024. (**Accepted**).

[7] Thanh Nguyen Canh, Huy-Hoang Ngo, Xiem HoangVan, and Nak Young Chong, "Toward Integrating Semantic-aware Path Planning and Reliable Localization for UAV Operation," *The 24th International Conference on Control, Automation and Systems (ICCAS)*, 2024. (**Accepted**).

[8] Thanh Nguyen Canh, Minh DoNgoc, Truong Nguyen Quang, Huong Bui Thanh, and Xiem HoangVan, "Underwater Image Enhancement for Depth Estimation via Various Image Processing," *The International Conference on System Science and Engineering (ICSSE)*, 2024. http://doi.org/10.1109/ICSSE61472.2024.10608973.

[9] Manh Do Duc, Thanh Nguyen Canh, Minh DoNgoc, and Xiem HoangVan, "Fusion LiDAR-Inertial-Encoder data for High-Accuracy SLAM," *International Conference on Mechatronic, Automobile, and Environment Engineering (ICMAEE)*, 2024. https://doi.org/10.48550/arXiv.2407.11870.

[10] Thanh Nguyen Canh, and Xiem HoangVan, "Machine Learning-Based Malicious Vehicle Detection for Security Threats and Attacks in Vehicle Ad-Hoc Network (VANET) Communications," *International Conference on Computing and Communication Technologies (RIVF)*, 2023, pp. 206–211. https://doi.org/10.1109/RIVF60135.2023.10471804.

[11] Thanh Nguyen Canh, Truong Son Nguyen, Cong Hoang Quach, Xiem HoangVan, and Manh Duong Phung, "Multisensor Data Fusion for Reliable Obstacle Avoidance," *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, 2022, pp. 385–390. https://doi.org/10.1109/ICCAIS56082.2022.9990495.

Awards

- Best Paper Award, IEEE International Conference on Research, Innovation and Vision for the Future (RIVF), December 2023.
- Science and Technology Scholarship from Vingroup Innovation Foundation, 2023.
- Best Paper Award, REV-ECIT Radio and Electronics Association of Vietnam, 2022.

References

[1] J. W. Goh, "Visual localization and mapping for indoor drone navigation," Ph.D. dissertation, Nanyang Technological University, Singapore, 2024.

[2] B. Alsadik and S. Karam, "The simultaneous localization and mapping (slam)-an overview," *Journal of Applied Science and Technology Trends*, vol. 2, no. 02, pp. 147–158, 2021.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[4] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 941–957, 2010.

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[6] W. Shi, J. Xu, D. Zhu, G. Zhang, X. Wang, J. Li, and X. Zhang, "Rgb-d semantic segmentation and label-oriented voxelgrid fusion for accurate 3d semantic mapping," *IEEE transactions on circuits and systems for video technology*, vol. 32, no. 1, pp. 183–197, 2021.

[7] T. N. Canh, A. Elibol, N. Y. Chong, and X. HoangVan, "Object-oriented semantic mapping for reliable uavs navigation," in *2023 12th International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, 2023, pp. 139–144.

[8] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[9] N. Aharon, R. Orfaig, and B. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*. Ieee, 2008, pp. 1–8.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2014.

[13] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[19] J. Lafferty, A. McCallum, F. Pereira *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Icml*, vol. 1, no. 2. Williamstown, MA, 2001, p. 3.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[21] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[23] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[24] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4151–4160.

[25] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 405–420.

[26] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[27] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[28] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[29] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[30] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[31] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[32] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[33] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[34] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[35] Z. Teed and J. Deng, "Deepv2d: Video to depth with differentiable structure from motion," *arXiv preprint arXiv:1812.04605*, 2018.

[36] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[37] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE transactions on robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[38] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 134–144.

[39] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[40] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[41] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph." in *Robotics: science and systems*, vol. 11. Rome, Italy, 2015, p. 3.

[42] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[43] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.

[44] F. Dellaert, D. J. Bruemmer, and A. C. C. Workspace, "Semantic slam for collaborative cognitive workspaces." in *AAAI Technical Report (5)*, 2004, pp. 85–86.

[45] L. Ma, J. Stückler, C. Kerl, and D. Cremers, "Multi-view deep learning for consistent semantic mapping with rgb-d cameras," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 598–605.

[46] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

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

[48] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4471–4478.

[49] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2018, pp. 10–20.

[50] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[51] S.-C. Wu, K. Tateno, N. Navab, and F. Tombari, "Scfusion: Real-time incremental scene reconstruction with semantic completion," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 801–810.

[52] W. Deng, K. Huang, X. Chen, Z. Zhou, C. Shi, R. Guo, and H. Zhang, "Semantic rgb-d slam for rescue robot navigation," *IEEE Access*, vol. 8, pp. 221 320–221 329, 2020.

[53] L. Li, Z. Liu, Ü. Özgïner, J. Lian, Y. Zhou, and Y. Zhao, "Dense 3d semantic slam of traffic environment based on stereo vision," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 965–970.

[54] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.

[55] D. Bersan, R. Martins, M. Campos, and E. R. Nascimento, "Semantic map augmentation for robot navigation: A learning approach based on visual and depth data," in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. IEEE, 2018, pp. 45–50.

[56] F. Y. Setiono, A. Elibol, and N. Y. Chong, "A novel room categorization approach to semantic localization for domestic service robots," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2021, pp. 1166–1171.

[57] D. Wu, B. Xie, and C. Tao, "3d semantic vslam of dynamic environment based on yolact," *Mathematical Problems in Engineering*, vol. 2022, 2022.

[58] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*. IEEE, 2018, pp. 32–41.

[59] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic slam based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75 545–75 559, 2018.

[60] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[61] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[62] A. Asgharivaskasi and N. Atanasov, "Semantic octree mapping and shannon mutual information computation for robot exploration," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1910–1928, 2023.

[63] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[64] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[65] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern recognition letters*, vol. 30, no. 2, pp. 88–97, 2009.

[66] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.