Multi-Level Semantic SLAM

by

Karter Krueger

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Masters of Science

in

Robotics Engineering Department

by

May 2023

APPROVED:

Professor Jing Xiao, Advisor

Professor Nitin Sanket, Thesis Committee Member

Professor Carlo Pinciroli, Thesis Committee Member

Abstract

This thesis presents a novel Multi-Level Semantic Simultaneous Localization and Mapping (ML Semantic SLAM) method designed to improve the accuracy and reliability of robot mapping and localization in sparse and repetitive environments. By incorporating both low-level and high-level semantic features, our approach addresses the challenges associated with limited distinguishable keypoint ORB features and generates a more semantically rich map. We detail the development, implementation, and evaluation of our ML Semantic SLAM system in various simulated and real-world environments, demonstrating its superior performance compared to traditional SLAM techniques, such as ORB-SLAM2. The system achieves up to a 70% error reduction in highly sparse environments and exhibits modest improvements in more moderate environments, showcasing its robustness and versatility. We also propose several future directions to extend the research. By continually refining and expanding upon the Multi-Level Semantic SLAM method, we hope to enable more accurate and reliable SLAM systems for various real-world applications.

Acknowledgements

I would like to thank those who have guided and supported me during my academic journey as they helped make this thesis possible.

I express my gratitude to my thesis advisor, Dr. Jing Xiao, for her expertise, support, and direction throughout the development of this thesis. Thank you as well to my committee members Dr. Nitin Sanket and Dr. Carlo Pinciroli for their time and support.

I thank my family and my mom, Karla, a professor whose hard work and dedication to academia was a source of my work ethic and inspiration to pursue graduate school. Additionally, I want to thank my sister, Karris, for her support during our time together at WPI. In loving memory of my late grandmother, Verla, I want to acknowledge her support and enthusiasm for robotics and engineering.

I am grateful for my undergraduate professors, Dr. Yan-Bin Jia and Dr. Alex Stoychev, who had an impact on my professional growth and my inspiration to pursue graduate school.

Special thanks go to my high-school FIRST Tech Challenge robotics mentors, Dave Mundhenke, Neil Erickson, and Val Frey, for their pivotal role in introducing me to the world of robotics, nurturing my passion for the field, and providing me with extensive hands-on robotics experience.

Lastly, I would like to extend my appreciation to my partner Trisha for her continuous support throughout this journey.

This work is partially supported by funding from Raytheon Technologies through the NSF Industry/University Cooperative Research Center (I/UCRC) on Robots and Sensors for Human Well-being (ROSE-HUB), NSF-1939061.

Contents

List of Figures

Chapter 1

Introduction

The past few decades have witnessed an exponential growth of robots and automation, reaching nearly every aspect of our society. As the world increasingly embraces new technologies, the use of autonomous robots has expanded to include a wide range of applications, from household chores and manufacturing to rescue operations and inspection. A core component of robotics development is the ability for these autonomous systems to perceive and navigate in a variety of environments. This ability, commonly referred to as Simultaneous Localization and Mapping (SLAM), has become a key component in the continued functionality and expansion of robotics.

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics, involving the estimation of a robot's pose within an unknown environment while building a map of the environment at the same time. It is a critical capability for autonomous systems, enabling them to perceive, navigate, and interact with their surroundings. This chapter provides a comprehensive introduction on the background of SLAM, including the mathematical formulation, key approaches, and common sensors used in SLAM.

One of the earliest works on robot SLAM was by J.J. Leonard et. al [18] with

a robot that used a servo-mounted range sensor to build a map and localize. Over time, SLAM continued to expand into new sensor domains and to 3D environments. Today, SLAM is most commonly performed using 3D LiDAR scanners and cameras. Visual SLAM has proved to be one of the most popular areas as cameras (monocular, stereo, and RGBD) are much lower cost than LiDAR scanners. SLAM is very widely used across applications in robotics and autonomous vehicles today for a variety of tasks such as indoor navigation, self-driving cars, and robotic exploration.

1.1 Mathematical SLAM Formulation

The idea of SLAM is that given a sequence of sensor measurements, the goal is to estimate the robot's pose and a map for the surrounding environment. The measurements are in a variety of forms such as IMU accelerations, camera features, and LiDAR points. The pose for 3D environments is typically defined in 6-DOF (degrees of freedom) space, with a position and orientation for the x, y, and z axes.

In the mathematical formulation, the robot's trajectory is represented by a sequence of poses, $x = x_1, x_2, ..., x_t$, where x_t denotes the robot's pose at the time step t. The map of the environment is represented by a set of features/landmarks, $m = m_1, m_2, ..., m_n$, where m_i denotes the i-th landmark in the map. The objective is to estimate the joint posterior distribution of the robot's trajectory and the map given a sequence of sensor measurements, $z = z_1, z_2, ..., z_t$, and optionally a sequence of control inputs, $u = u_1, u_2, ..., u_t$. Mathematically, the SLAM problem can be formulated to find x and m using z and u as follows:

$$P(x, m|z, u)$$

To solve this problem, we need to model both the motion model and the observation

model. The motion model represents how the pose x of the robot changes over time due to the control inputs u, expressed as:

$$x_t = f(x_{t-1}, u_t) + w_t$$

where f is a nonlinear function representing the dynamics of the robot with w_t motion noise.

The observation model relates the sensor measurements z to the pose x of the robot and the mapped landmarks m, expressed as:

$$z_t = h(x_t, m) + v_t$$

where h is a nonlinear function representing the sensor observations of the landmarks with v_t measurement noise.

1.2 Key SLAM Approaches

There are several approaches to solving the SLAM problem, with the most popular methods based on either filtering or optimization techniques.

Filtering methods, such as particle filters and Kalman filters, recursively estimate the robot's pose and the map based on the motion and observation models. Particle filters represent the posterior distribution using a set of particles and update them using importance sampling, while Kalman filters maintain a Gaussian distribution over the robot's pose and the map and update it using linearizations of the motion and observation models.

Graph-SLAM formulates the SLAM problem as a graph optimization problem, where the nodes of a graph represent robot poses and landmarks/features, and the

edges represent the constraints between nodes created by the motion and observation models. The objective is to find the graph of nodes that minimizes the error of the edge constraints. This method is especially used in applications that require loop-closing on the map, as it allows for efficient and accurate estimation of the robot's trajectory and the map. There are various techniques to solve the graph optimization problem in Graph-SLAM, including iterative linearization methods, such as Gauss-Newton and Levenberg-Marquardt algorithms, and sparse linear solvers, such as the Conjugate Gradient method. Additionally, there are several methods for loop closure detection in Graph-SLAM, including appearance-based techniques, which rely on visual similarities between images, and geometric methods, which exploit the spatial consistency of features. In this work, we focus on the graph approach to SLAM, as it provides a flexible and efficient framework for incorporating our novel semantic SLAM method.

1.3 Common Sensors for SLAM

There are several common sensors used in SLAM, including inertial measurement units (IMUs), LiDARs, and cameras.

1. IMUs: Inertial Measurement Units measure the linear acceleration and angular velocity of a robot, providing information about its motion. They are often used as a supplementary sensor that is used in combination with other sensors to provide more accurate and robust pose estimation.

2. LiDAR: Light Detection and Ranging (LiDAR) sensors use laser beams to measure distances to objects in the environment, creating high-resolution 2D or 3D point clouds. These sensors are particularly useful for mapping and

obstacle detection, especially in high-precision applications, but are often expensive and require more maintenance.

3. Vision: Cameras, including monocular, stereo, and RGB-D systems, capture visual information about the environment. Cameras have the advantage of being low-cost, lightweight, and easy to use compared to other sensors. Monocular cameras are typically the goal sensor as they are the cheapest and easiest to use, but they are the most challenging to work with, due to the lack of depth information that is used to determine the scale of the SLAM map. Stereo and RGB-D (color and depth) cameras are more popular when map scale and higher-precision are required. Cameras are also widely used as they provide rich visual information that can be used for semantic understanding of the environment, which is essential for our proposed ML Semantic SLAM method.

In recent years, SLAM research has primarily focused on vision-based methods due to the accessibility and simplicity of cameras.

1.4 Vision-based SLAM

Vision-based SLAM methods rely on cameras to obtain observations of the environment. These methods can be categorized into feature-based, direct, and semi-dense approaches, depending on the type of visual information they exploit. Feature-based methods extract and match distinct visual features, such as points or lines, from successive frames to estimate the robot's motion and reconstruct the environment. Direct methods, on the other hand, use all available pixel intensities to estimate motion and depth directly. Semi-dense approaches lie in between these two extremes, utilizing a subset of the image pixels to estimate depth and motion.

The trade-offs in accuracy and computational cost are between these approaches are typically weighed based on what processing power is available on-board the robot and what performance is required for the application.

1.5 Semantic SLAM

Semantic SLAM extends traditional SLAM methods by incorporating high-level semantic information extracted from the sensor data, typically using machine learning techniques. This additional information can help improve the accuracy and robustness of the SLAM system, as well as generate more informative maps. Semantic SLAM methods can be classified into several categories based on the type of semantic information used, such as object-level semantics, place-level semantics, or scene-level semantics. One common example of an application for Semantic SLAM is robots tasked with searching for specific objects in an environment, such as a house.

Chapter 2

Visual and Semantic SLAM Methods

In the following sections, we categorize existing related visual SLAM methods based on the types of features they use.

2.1 Point-based SLAM

In both the early and recent methods of visual SLAM, visual keypoint features were found using various methods that detect pixels with high local contrast scores, such as SIFT [20] and ORB [28]. Many SLAM approaches are built purely on these keypoints.

ORB-SLAM2 [23] is one of the best known SLAM methods in recent years, which is capable of running on monocular, stereo, or RGB-D cameras. Visual features are found using ORB [28] points that can be matched between frames based on unique feature descriptors. Keypoints without depth cannot be used for determining scale, but contribute to rotation and translation estimations. The camera pose is estimated

using Bundle Adjustment (BA) to optimize orientation and position by minimizing re-projection error of the matched keypoints in global 3D coordinates.

2.2 Line-based SLAM

In one alternative to using point features, Chandraker et al. [4] used lines to perform better in scenes that lack reliable point features, but have an abundance of line features, such as offices and buildings. Infinite lines are tracked to avoid the issue of determining end-points. Straight lines are found with a Sobel edge detector and tracked using multi-level optical-flow. A RANSAC-based framework is then used to estimate position without using bundle adjustment. Another example of a line-based SLAM [35] targets corridor environments by tracking vertical and floor lines. SLAM is performed on the line features using an EKF-SLAM method to determine the line positions and camera pose. While similar in features, [13] uses a more modern approach to SLAM, with a bundle adjustment (BA) back-end based on both points and edgelets (short line sections). Edgelets are measured by changes in pixels along the perpendicular and parallel directions.

2.3 Semantic SLAM

Semantic SLAM approaches fall into two categories, some use semantic objects purely for creating a rich map and others use the semantics for both the mapping and robot localization. In work from Qian et al. [26], semantic objects are detected using YOLOV3 [27]. ORB features are associated to the enclosing object bounding boxes to be used for object data-association. Quadrics are then fitted around the features of each object. An extension of this [25] also used the semantic objects to perform more accurate loop closure in repetitive environments, but did not focus on

improving localization in such environments.

In [9], Guan et al. developed a semantic Point-Object SLAM (PO-SLAM) that considered both points and objects for localization and mapping. PO-SLAM focuses on improving SLAM accuracy in settings with variations of appearance and brightness in the environment. The tracking is based on point-point, point-object, and object-object data associations. Point features use ORB descriptors and point-point association from ORB-SLAM2 [23]. Object features are extracted from images using the YOLOv3 object detector and objects are segmented from the background using a depth histogram from the RGB-D camera. Point-object associations are then determined using point features that are located within the segmented object boundaries. Object-object associations are determined between frames by distance between object coordinates and classes. Bundle adjustment is performed by combining the point-point, point-object, and object-object errors.

CubeSLAM [33] performs SLAM using cube objects, as an extension on ORB-SLAM2 [23]. Cuboid proposals are estimated from 2D bounding boxes using vanishing points. Bundle adjustment (BA) is performed on new object measurements to optimize camera, object, and point poses. Point-features are included, as objects alone typically aren't enough to fully constrain the camera poses. Dynamic objects are separately included with motion properties in the BA to enhance tracking.

[34] uses both objects and planes for SLAM in structured environments. Objects are assumed to be structured as they are supported by the floor and match nearby wall normals. Data association then matches objects and planes between frames by finding the match with the highest number of identical points contained in the object or along the plane. Standard bundle adjustment then jointly optimizes the poses of the camera and all objects.

In addition to SLAM, there are several non-SLAM works that have considered

semantic features for localization without the mapping component. One example, Visual Semantic Odometry [19], is incorporates semantic labels to the low-level feature points to improve the matching accuracy and therefore the odometry accuracy. This is primarily useful for distant features that look similar, but belong to separate semantic classes. Another example, SimVOIDS [12], is focused on a pure neural-network approach for predicting semantic objects, depth estimation, and pose estimation from a series of RGB images.

2.4 Notable SLAM Methods

ORB-SLAM [3, 22, 23] is one of the most popular visual SLAM methods used today due to it's robust performance and open-sourced code. There are now three versions of the ORB-SLAM method that have iteratively built on the previous versions to improve performance and add additional features. ORB-SLAM2 is still very widely used as it supports monocular, stereo, and RGBD sensing modes and the work in this thesis is built on top of an ORB-SLAM2 framework due to it's RGBD support.

RTAB-MAP [17] is another example of an open-source SLAM method that is widely used. The primary advantage of RTAB-MAP is that it outputs a dense 3d map of the environment rather than a sparse set of feature points like ORB-SLAM2. Thanks to the dense map, RTAB-MAP is more applicable to applications that require a 3D occupancy map for obstacle avoidance.

2.5 Limitations and Challenges

While there are many versions of existing SLAM methods that cover a wide variety of specialties, there exist environments where nearly all SLAM methods struggle and these are areas that either lack features or contain only repetitive features. This is

a fundamental struggle for SLAM methods as they rely on unique, distinguishable features that can be matched between image frames.

Chapter 3

Problem Statement and Approach

Despite years of advancements, SLAM algorithms often struggle in environments with scarce distinctive landmarks or features. These challenging environments necessitate the development of novel SLAM approaches that can overcome the limitations of conventional methods. One such application with significant potential for improved safety and efficiency is the aviation industry, where autonomous robots can streamline the inspection process for cracks and damages in airplanes.

Human inspection of aircraft is a labor-intensive and time-consuming process, requiring highly skilled technicians to carefully examine every surface of an airplane, searching for defects or signs of wear. By employing autonomous robots in aircraft inspection, the frequency and reliability of these inspections can be increased, ultimately leading to safer air travel. Airplanes serve as a compelling edge case for developing a novel SLAM method, given their highly repetitive interior textures and limited variety of displayed objects.

3.1 Problem Statement

Conventional SLAM approaches, primarily based on vision or RGB-D sensing, when applied to repetitive and sparse environments lacking unique objects and textures, often leads to the accumulation of errors, potentially causing navigation failures in environments such as airplane interiors. Consequently, autonomous robots tasked with inspecting airplanes for defects or damages may experience challenges in accurately mapping and navigating the environment, limiting their ability to identify potential damage and ensure aircraft safety.

The shortcomings of existing methods in these challenging environments primarily lie in what types of features are used. Existing methods have mostly focused on using a single type of feature per method, being ORB feature points, lines/edges, and semantic objects. There has been some combinations of these features, but not to the fullest extent that would fully support such a challenging environment such as an airplane.

3.2 Approach of This Thesis

To address these challenges, this thesis presents the development of a Multi-Level Semantic SLAM (ML Semantic SLAM) method, leveraging both high-level semantics and low-level key-point features in RGB-D sensing. By integrating machine learning techniques for semantic segmentation and object detection, our proposed approach achieves superior performance in tracking the robot's position and orientation while constructing a semantically-detailed map of the environment. This capability enables the robot to navigate and perform inspection tasks with increased precision and reliability.

The primary contribution of this thesis is the combination of all of these types of

existing features (points, lines, and semantic objects), in addition to incorporating additional features such as semantic regions. The combination of all of these features together is what sets our approach out from the others to perform the best in extremely sparse and repetitive environments.

Chapter 4

Multi-Level Semantic SLAM

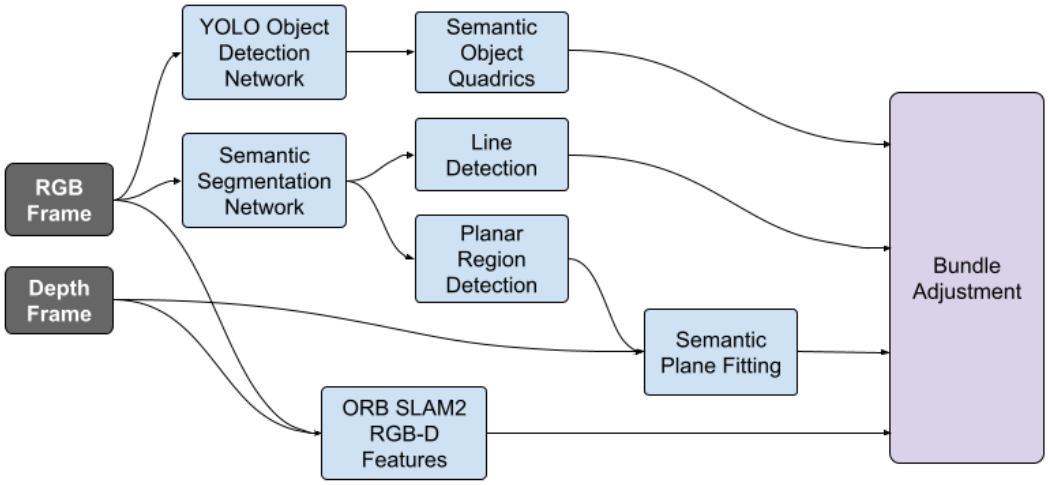

Figure 4.1: System diagram of feature extraction leading to BA

In this chapter, we delve into the technical details of our Multi-Level Semantic SLAM (ML Semantic SLAM) method, which aims to address the challenges of conventional SLAM approaches in sparse and repetitive environments by leveraging multiple levels of high-level objects and low-level features. Our ML Semantic SLAM builds upon the well-established ORB-SLAM2 framework and incorporates additional semantic information, such as objects and planar surfaces, into the bun-

dle adjustment (BA) optimization process. This fusion of high-level and low-level features allows the system to achieve improved localization accuracy and robustness, as demonstrated in our experimental results in Chapter 4.

The structure of this chapter is as follows: First, we provide an overview of the ML Semantic SLAM system architecture and discuss its main components. Next, we present the details of our feature extraction process, which includes the detection of point features, line segments, and objects, as well as the identification of planar surfaces. We then explain how these extracted features are integrated into the bundle adjustment module to optimize the SLAM solution and produce a semantic map.

4.1 System Architecture

As shown in Fig. 4.1, our ML Semantic SLAM system consists of several interconnected modules that work in parallel to process the input data, extract features, and optimize the graph with Bundle Adjustment at the end. This architecture method builds upon the ORB-SLAM2 framework [23] and further extends it with additional semantic capabilities introduced by [26]. The main components of the system include:

- Low-level point feature tracking: We employ ORB-SLAM2 to track low-level point features, which are extracted using the ORB feature detector. This module is also responsible for maintaining the continuity of the tracked features across consecutive frames.

- Semantic Segmentation: We extract class labels for each pixel and use these as part of the following features:

  - Line segment detection: Line segments are detected on the semantically-segmented frames to identify edges along semantic objects and class boundaries. This step enhances the geometric representation of the environment and provides additional constraints for the optimization process.
  - Planar surface extraction: Planar surfaces, such as walls, floors, and ceilings, are identified from the semantic masks in conjunction with the depth information obtained from an RGB-D camera. These planar surfaces provide valuable geometric information for the SLAM optimization process.

- Object detection and segmentation: Objects within the environment are detected using both the semantic masks and YOLOv3 [27]. The combination of these methods allows us to accurately identify and track individual objects and their boundaries.

- Bundle adjustment module: All the detected features, including point features, line segments, objects, and planar surfaces, are integrated into the bundle adjustment module. This module optimizes the positions of all objects, features, and key-frames by minimizing the re-projection error in 3D space.

The above features are annotated on an example image in Fig. 4.2. Next, we explain the full details for all of the above components.

4.2 Multi-Level Feature Extraction

In this section, we present the details of our multi-level feature extraction process, which involves the detection of point features, line segments, objects, and the identification of planar surfaces. We discuss how these extracted features contribute to

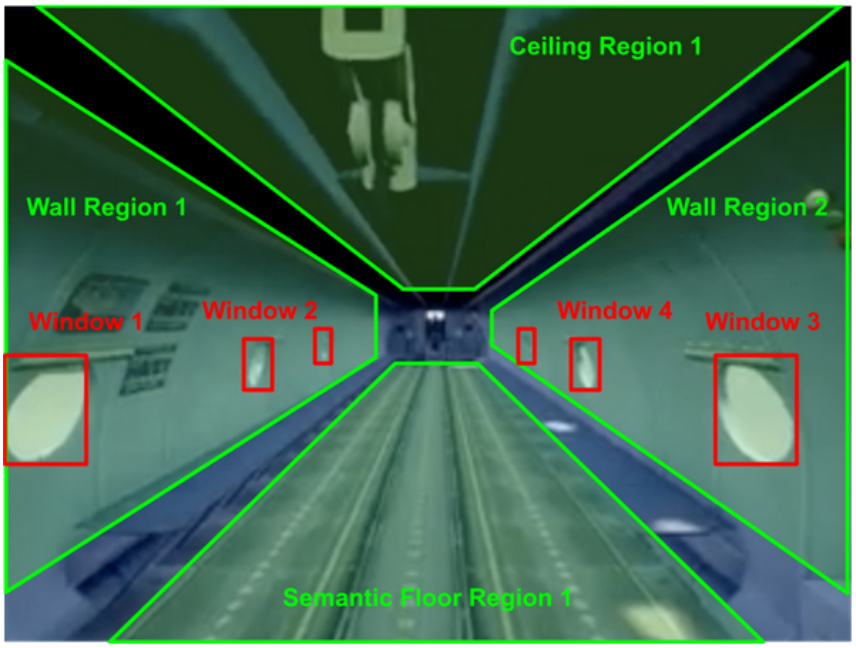

Figure 4.2: A semantic map that represents what is seen by the Multi-Level Semantic SLAM, showing features of multiple levels in the airplane environment

the robustness and accuracy of our ML Semantic SLAM approach and provide a comprehensive understanding of the overall system.

4.2.1 Semantic Segmentation

Semantic segmentation plays a crucial role in our ML Semantic SLAM approach, as it provides the foundation for extracting high-level semantic information from the environment. Semantics of the scene often extend beyond objects, as semantic regions stretch beyond the view of the camera. Given that regions such as walls, floors, and ceilings typically stretch across many views, they can be incorporated into SLAM as consistent landmarks to help localize, and as additional map information the robot can use to interact with the environment.

In this sub-section, we describe our custom fine-tuned DeepLabV3 RESNET-50 architecture and its application to different environments, such as airplane interiors

and general indoor scenes. We also discuss the training process for the segmentation network and illustrate the quality of the semantic predictions obtained from this network.

DeepLabV3 RESNET-50 Architecture

Our semantic segmentation is based on the DeepLabV3 RESNET-50 architecture, which performs pixel-wise semantic segmentation on RGB images. We have adapted the original network to produce 13-channel output, representing 13 classes specific to our target environments, such as walls, floors, ceilings, windows, and doors, with one class reserved for unknowns. Fig. 4.3 shows an example of the input RGB image and the corresponding semantic class predictions.

Training and Fine-tuning for Airplane Environment

To tailor the segmentation network to the airplane environment, we started with pre-trained weights for the COCO dataset and then fine-tuned it on objects commonly found in aircraft environments, such as windows and doors. Pairs of RGB and ground truth segmentation masks for the fine-tuning process were collected in simulated aircraft environments using the UnReal Engine [6].

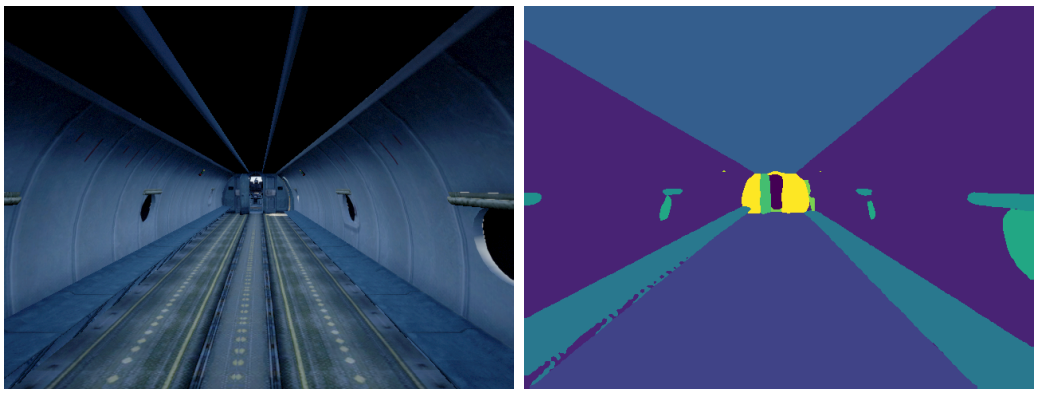

Figure 4.3: RGB image (left) with semantic class predictions (right) for an airplane environment

Semantic Segmentation for General Indoor Environments

For general indoor environments, we utilized a standard Semantic Segmentation network from Zhou et al. [36, 37] with weights pre-trained on the ADE20k dataset of indoor scenes. This network is suitable for a large variety of indoor settings, such as bathrooms, hallways, offices, and kitchens. It outputs semantic labels for objects such as rugs, toilets, sinks, and cabinets, in addition to general classes of floors, walls, and ceilings. An example view of the corresponding RGB input and semantic segmentation output frames for a bathroom environment is shown in Fig. 4.4.

Figure 4.4: RGB image (left) with semantic class predictions (right) for a bathroom environment

By leveraging semantic segmentation, our ML Semantic SLAM system is capable of extracting high-level information about the environment, which is essential for accurately detecting and tracking objects and planar surfaces. This additional layer of semantic understanding enables our approach to track higher-level semantic regions that are easier to track in repetitive scenes compared to conventional low-level features.

4.2.2 Semantic Line Detection

In this sub-section, we detail the process of semantic line detection, an essential component of our approach. We explain how semantic lines are extracted from the scene using semantic masks and present the data association algorithm used for tracking the detected line features.

Semantic line detection serves as a valuable tool in SLAM applications, as it helps in identifying and tracking structures in environments with little texture but distinct boundaries. Such environments include indoor spaces, where lines can be found along the seams of walls, ceilings, and floors, or between sparse objects. In this sub-section, we discuss the line detection process and the data association algorithm used to track these lines across frames.

Line Detection from Semantic Masks

We start by iterating through the semantic class masks and perform contour detection using the algorithm proposed in [32] to find clusters of pixels representing individual windows, walls, or other structures. Next, we filter the contours and discard any contour with a region area smaller than a preset threshold. To simplify the region boundaries, we compute the convex hull for points in the contour region using Sklansky's algorithm [31], which runs in $O(N \log N)$ complexity, where N is the number of pixels on the contour. The convex hull is then approximated by a simpler polygon using the Douglas-Peucker algorithm [5]. We consider the line segments on these polygons as semantic line segments. An example of the detected edges and their endpoints is shown in Fig. 4.5.

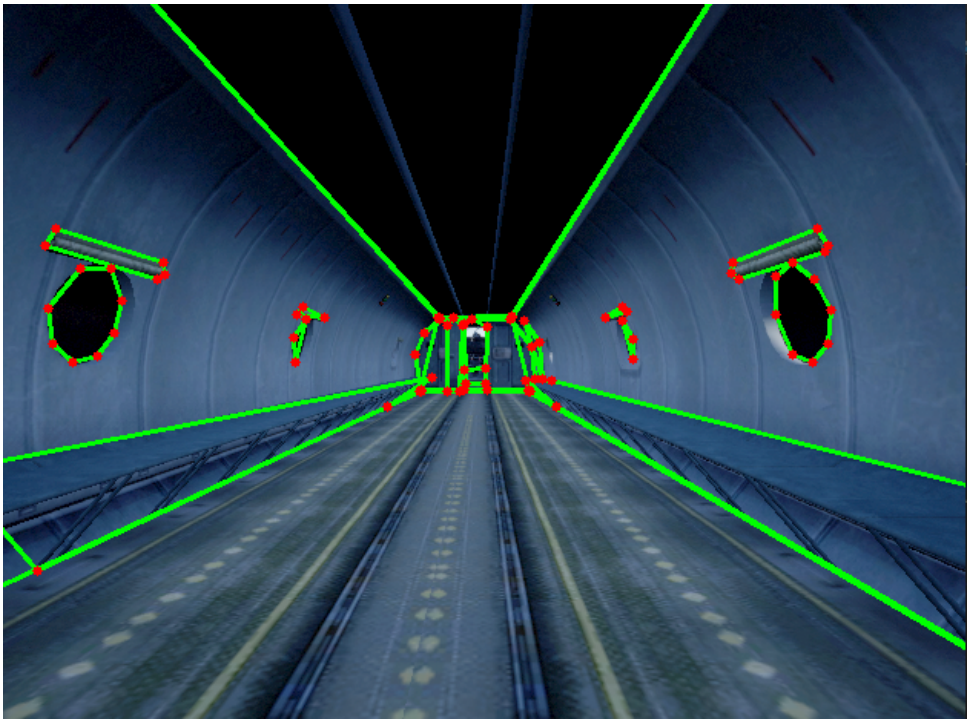

Figure 4.5: Line segment detection based on semantic boundaries, demonstrated in the airplane environment

Data Association for Semantic Lines

We formulate the data association problem for semantic lines as a maximum weighted bipartite matching problem. For each pair of semantic lines l_m and l_n, we introduce a set of Boolean decision variables:

$$s_{mn} = \begin{cases} 1, & \text{if } l_m \text{ is matched with } l_n, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

For every line segment of the same class, a score is calculated between every semantic line segment in $Frame_a$ and every potential matching line segment from $Frame_b$. The score between line l_m in $Frame_a$ and l_n in $Frame_b$ is calculated as the weighted sum of the difference of endpoint coordinates as follows:

$$c_{mn} = w_1 \sum_{i=1}^{2} ||\mathbf{x}_{m,i} - \mathbf{x}_{n,i}|| \tag{4.2}$$

where w_1 is the weight for the endpoint coordinates difference.

The maximum weighted bipartite matching problem seeks to perform the following constrained optimization:

$$s_{mn} \sum_{l_m \in Frame_a} \sum_{l_n \in Frame_b} c_{mn} s_{mn} \tag{4.3}$$

$$\text{s.t.} \sum_{l_m \in Frame_a} s_{mn} \leq 1, \quad \sum_{l_n \in Frame_b} s_{mn} \leq 1. \tag{4.4}$$

The constraints (4.4) mean that each semantic line in $Frame_a$ can be assigned to at most one line in $Frame_b$ and vice versa.

The problem defined in 4.3 can be solved using a cost-scaling push-relabel algorithm [2, 8, 11], readily implemented in [24]. This algorithm has a time complexity of $\mathcal{O}(\sqrt{N}M \log(NC))$, where N and M are the number of nodes and edges in the bipartite graph, and C is the largest edge cost (all edge costs need to be converted to integers).

By incorporating the semantic line detection and data association methods explained above, our system can effectively track semantic line features in environments with limited texture or distinct structures, such as indoor spaces. This tracking contributes to the overall accuracy and robustness of our SLAM system in a variety of environmental contexts.

4.2.3 Plane Detection and Tracking

Planar surfaces are prevalent in various environments and often stretch across multiple views, making them valuable features to track. In this section, we explain the process of detecting and tracking planar surfaces, with the help of the Point Cloud Library (PCL) [29].

Plane Detection

To detect planes, we first generate binary masks for the wall, floor, and ceiling classes from the output of the semantic segmentation network. Masked pixels are then separated into clusters (e.g., the left and right walls are separate). Depth values from the RGB-D camera are collected for the clustered pixels, and the pixel (u, v) coordinates are projected to the 3D space of (x, y, z) using the intrinsic matrix of the calibrated camera. We then fit a plane to the 3D coordinates with PCL using least-squares minimization with RANSAC outlier-removal, which determines the optimal A, B, C, D parameters for the following plane representation:

$$Ax + By + Cz = D \tag{4.5}$$

The normal vector and intercept point of the plane are calculated and stored for subsequent matching.

Plane Tracking and Data Association

Data association matches the planes between frames by minimizing a plane-difference error composed of the normal vector and plane separation distance. This approach allows the module to consistently track planar surfaces across consecutive frames.

Figure 4.6 provides an example of fitted planes and detected windows in the

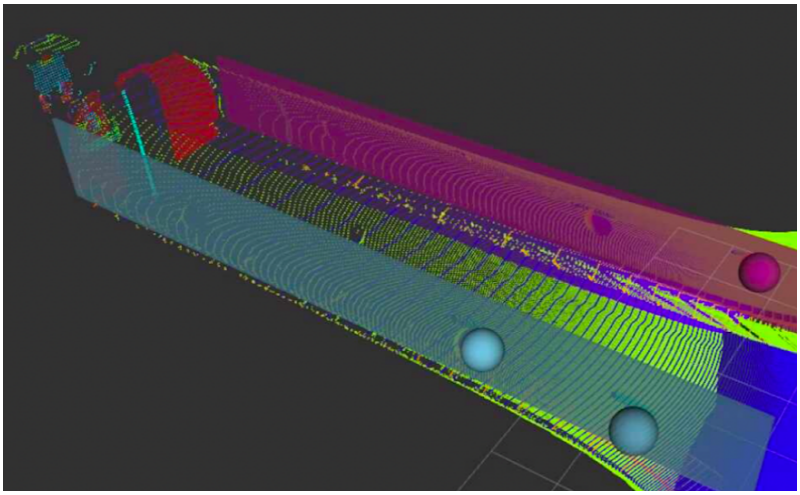

Figure 4.6: Fitted planes and several detected windows for the airplane environment are overlaid on the semantically-colored point cloud.

airplane environment, overlaid on the semantically-colored point cloud.

4.2.4 Object Detection and Tracking

The environment contains objects with varying dimensions, from flat 2D objects such as windows and paintings to 3D objects like chairs and tables. In this subsection, we describe the methods employed for detecting and tracking both types of objects using semantic segmentation and the YOLOv3 neural network.

2D Object Detection and Tracking

For flat 2D objects, we detect clusters of pixels in each class-specific semantic mask generated from the output of the semantic segmentation network. Additionally, we use the YOLOv3 [27] neural network to detect a higher number of unique object classes, as semantic segmentation networks typically focus on higher-level regions rather than individual objects. We used a YOLO model pre-trained on the COCO dataset and we fine-tuned it on a custom dataset for objects that are common to the airplane environment, such as windows.

The custom dataset consists of RGB images with bounding-box labels of window objects from various views, both interior and exterior, of simulated aircraft models. The network outputs a 2D bounding box, class ID, and confidence for each detected object in the scene. Bounding boxes are projected back to the camera plane, and 3D coordinates are determined using depth information. Semantic 2D objects are represented by rectangles due to their planar properties. Data association for semantic 2D objects is performed by matching objects with the same class and minimizing a score composed of position error, object area difference, and the intersect over union (IoU) of the two masked regions. Figure 4.7 provides an example of YOLO detecting 2D windows in an aircraft interior.

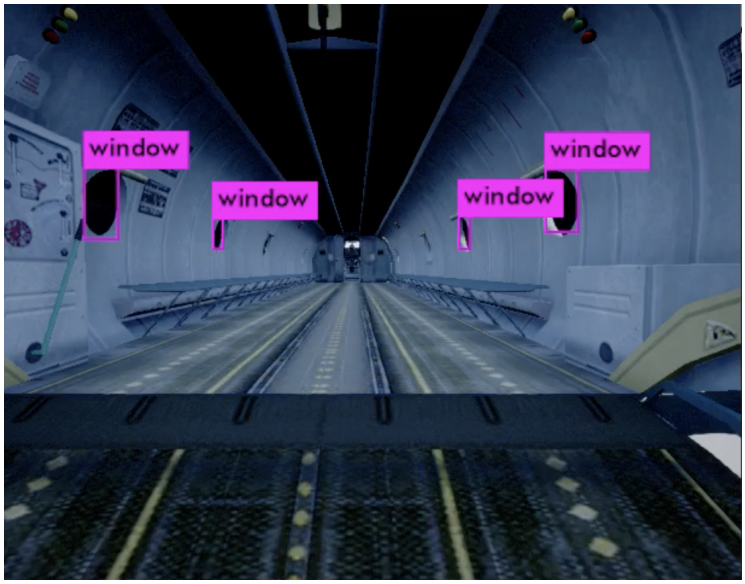

Figure 4.7: YOLO detections on the RGB frame

When operating in environments outside of airplanes, such as homes and offices, the original pre-trained weights for COCO can be used to detect many common objects. The standard model based on COCO includes common objects such as people, chair, table, computer, and etc. For a bathroom dataset that we also test on, there are common classes such as sink, towel, and bottle.

3D Object Detection and Tracking

Detection and tracking of 3D objects is based on the method presented in work by Qian et al. [26], where objects are represented as quadrics. The data association is conducted using bipartite matching to maximize a score based on the ORB features and BoW (bag of words) vectors located inside the bounding quadric region of the object.

The above object detection and tracking techniques effectively manage both 2D and 3D objects, allowing for a comprehensive understanding of objects in the environment.

4.3 Bundle Adjustment Formulation

Bundle Adjustment (BA) is a crucial optimization step in SLAM systems that aims to improve the consistency and accuracy of the estimated camera poses and map features. Given the initial estimates of camera poses and map features, BA refines the poses by minimizing the reprojection error between observed and predicted feature positions in the image plane. This optimization is performed by iteratively updating the camera poses and feature positions based on error minimization criteria until convergence. The bundle adjustment process leverages the redundancies in feature observations across multiple frames, allowing it to deliver a globally consistent map and camera trajectory. An example of the graph for bundle adjustment, with feature nodes and measurement edges is shown in Fig. 4.8.

4.3.1 Edge Definition and Covariance Weights

Edges are defined by an observation in camera coordinates and the 3D XYZ position of the point in global coordinates to be optimized. The edge observation values,

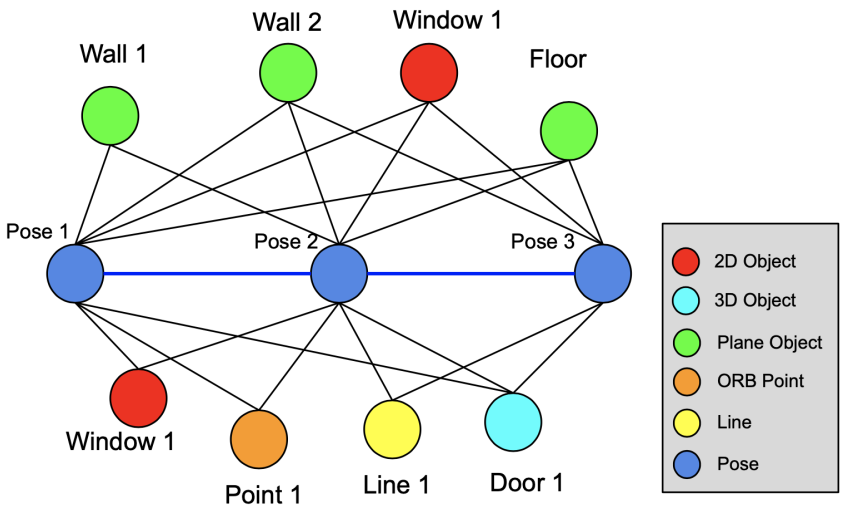

Figure 4.8: Factor Graph of Objects and Poses

$z = (u_l, u_r, v)$, are in camera image coordinates, with v being the "y-axis" of the image plane and the "x-axis" image plane coordinates of u_l and u_r being from cameras l and r, respectively. When a right camera is absent, such as the case where we have a single RGB-D camera, the u_r value is estimated using the corresponding depth value d and a constant stereo-camera baseline value bl as

$$u_r = u_l - bl * d$$

Edge covariance weights play a crucial role in the accuracy of the SLAM system, as they should be proportional to the confidence of the measurement to ensure erroneous points do not skew positions during optimization. In feature-repetitive environments, the ORB features may provide misleading readings due to mismatches; thus, we multiply the covariance by a scale factor of 20 times the original values. The covariance values of the semantic features are weighted proportional to their cubed depth values, as the furthest features exhibit the most noise and are the hardest to properly estimate and track. We tuned these parameter weights and definitions to

find the appropriate balance based on the noise of the ORB feature points and the stability of the semantic features.

4.3.2 Optimization Formulation

The bundle adjustment aims to perform a joint maximum a posterior (MAP) estimation of map objects (2D objects, 3D objects, plane objects, ORB Points, lines) and camera poses for the keyframes that have been saved along the robot trajectory. In the optimization process, $p(q_{i+1} \mid x_{i+1}, x_i)$ represents the probability of the odometry measurement given the current and next camera poses, $p(x_i)$ and $p(\tau_j)$ are the priors for camera poses and map objects, respectively, and $p(z_i^j \mid x_i, \tau_j)$ denotes the probability of the reprojection error of the map object τ_j observed in keyframe K_i given the camera pose x_i and the map object itself. A mathematical formulation for the optimization process is below:

$$\begin{aligned} \mathcal{X}^*, \mathcal{T}^* = {}_{\mathcal{X},\mathcal{T}} & \underbrace{\prod_{x_i} p(q_{i+1} \mid x_{i+1}, x_i)}_{\text{Odometry}} \cdot \underbrace{\prod_{\tau_i} p(x_i)}_{\text{Pose Prior}} \cdot \\ & \underbrace{\prod_{\tau_j} \prod_{x_i} p(z_i^j \mid x_i, \tau_j)}_{\text{Map object reprojection error}} \cdot \underbrace{\prod_{\tau_j} p(\tau_j)}_{\text{Object Prior}} \end{aligned} \tag{4.6}$$

where $\mathcal{T} = \tau_j$ is the set of map objects, $\mathcal{X} = x_i = T_w^i$ is the set of camera poses of keyframes, q_i is the odometry measurement, and z_i^j is the measurement of object τ_j on keyframe K_i. By assuming Gaussian measurement and process models and a uniform distribution of object and pose, taking the negative log on the objective function in (4.6) can rewrite the problem as a nonlinear least-squares problem:

$$\begin{aligned}\mathcal{X}^*, \mathcal{T}^* =& \underset{\mathcal{X},\mathcal{T}}{} \sum_{x_i} \| h_o(x_{i+1}, x_i) - q_{i+1} \|^2_{\Sigma_u} \\ &+ \sum_{\tau_j} \sum_{x_i} \| h_s(x_i, \tau_j) - z_i^j \|^2_{\Sigma_z}\end{aligned} \tag{4.7}$$

where $\|\cdot\|_\Sigma$ is the Mahalanobis norm, and Σ_u and Σ_z are the covariance matrices of odometry measurements and map object measurements, respectively. Note that Σ_z is not a constant covariance matrix; instead, it should be weighted according to the quality of the measurement, as discussed earlier. The values h_o and h_s are the sensor models of odometry and map objects, respectively.

The nonlinear least-squares problem described in (4.7) can be solved using modern optimization libraries such as g2o [16]. These libraries provide efficient optimization techniques tailored to SLAM systems, handling large-scale optimization problems with many poses and features. Other popular solvers include Ceres Solver [1] and the iSAM2 algorithm [10].

4.4 Chapter Summary

In this chapter, we have presented a comprehensive overview of the Multi-Level Semantic SLAM method. The proposed approach combines deep learning and geometric techniques to improve the performance of traditional SLAM systems. We discussed the system architecture and detailed all of the key components that take place in the feature extraction process and the bundle adjustment.

The Multi-Level Feature Extraction section provided an extensive look into the process of acquiring semantic information and detecting different types of features in the environment. Semantic segmentation is the foundation for feature extraction,

as it allows for the identification and classification of objects and structures. We also went into detail for the various sub-sections of this process, such as semantic line detection, plane detection and tracking, and object detection and tracking. By incorporating multiple feature representations, we improve the robustness of the SLAM system, especially for environments that lack features. Our bundle adjustment formulation is responsible for refining the estimated camera poses and map features using all of the available semantic information and features.

In conclusion, the Multi-Level Semantic SLAM method presented in this chapter showcases the potential of combining deep learning techniques with traditional geometric SLAM approaches. The next chapter will validate the proposed approach through experimental tests, further demonstrating the benefits and effectiveness of the Multi-Level Semantic SLAM method in real-world scenarios.

Chapter 5

Simulation Environments and Experimental Evaluation

In this chapter, we evaluate the performance of our Multi-Level Semantic SLAM method using a variety of simulated and real-world environments. The goal of these experiments is to validate the performance of the proposed method and demonstrate its robustness under challenging conditions. This chapter also highlights a motivating example where existing methods failed to track and how our system fills the gap. The details of our simulated experimental setup are also highlighted to show how we constructed the environments.

5.1 Experimental Simulation Setup

First, we walk through the setup of our key simulation environments of the airplane for the application of airplane inspection for cracks and damage.

5.1.1 Airplane Exterior Simulation with Gazebo

We first started with a model of a common airplane exterior in the Gazebo [14] physics simulator. We simulated an aerial robot (drone) using the RotorS [7] library from ETHZ ASL. The RotorS library is capable of simulating a variety of multirotor aerial vehicles in Gazebo along with a variety of sensors including IMUs and RGB-D (or stereo) cameras. We utilized the firefly vehicle with an RGB-D camera setup for our testing. Visuals of both the simulated airplane (Fig. 5.1) and the simulated multirotor vehicle (Fig. 5.2) below.

Figure 5.1: Exterior of a simulated plane environment (with the drone present in the lower center)

5.1.2 Airplane Interior Simulation with Unreal Engine

As we expanded our inspection focus to airplane interiors, we realized the need for a simulation platform capable of higher-resolution modeling and textures, which Gazebo wasn't designed for. This led us to use Unreal Engine [6] and Microsoft Airsim [30] to simulate a high-resolution model with ability to control an aerial

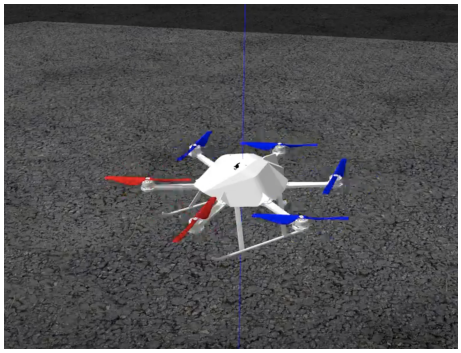

Figure 5.2: Simulated multirotor agent in Gazebo using the RotorS [7] library

robot (quad-rotor) through ROS and Python interfaces. The simulated interior environment is displayed in Fig. 5.3 from the perspective of the aerial robot camera.

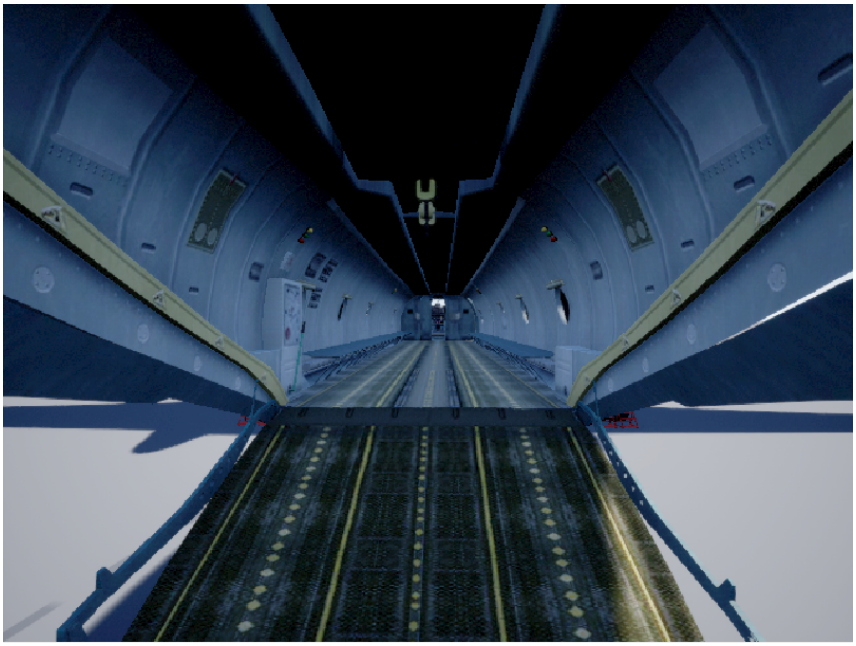

Figure 5.3: A long hallway-like interior of a simulated cargo plane environment

5.1.3 Augmented Simulation Model Potential

The environment also supports modifications such as adding obstacles and cargo to the simulation for more challenging navigation tasks and SLAM testing on additional object classes, as seen in Fig. 5.4.

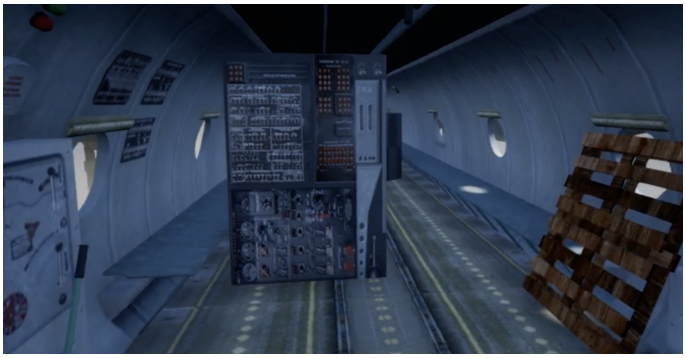

Figure 5.4: Plane interior simulation with added cargo and obstacles

Augmented models for more realistic textures from real airplane images. In one example, we modify the simulated textures using real-world airplane photos of textures that include real windows and riveted side panels. An example of the modified texture with patches of real images demonstrated in Fig. 5.5 and Fig. 5.6. This example is not fully complete as the textures could be blended further, but largely serves the purpose to show that augmenting the simulated models is possible for further realistic testing.

Figure 5.5: Plane exterior with a sample of augmented realistic textures included

We also experimented with creating more realistic simulation environments by reconstructing public videos of airplanes into a 3D model using Neural Radiance Fields (NeRFs) [21]. An example of an environment reconstructed from a video

Figure 5.6: Plane exterior with a sample of augmented realistic textures included

using a NeRF method is included in Fig. 5.7. We did not explicitly test our SLAM method in this environment, but show that it is another avenue to explore for more realistic SLAM testing in future works.

Figure 5.7: Plane model reconstructed using NeRF

5.2 Experimental Results and Comparisons

To test the effectiveness of our ML Semantic SLAM method, we first demonstrate results on an extremely sparse environment, a simulated airplane interior. We evaluate the performance of our system on a simulated environment of the interior of an airplane with sparse objects and very repetitive texture features [1]. The key test

[1]Note that a real airplane could offer more features and thus a less challenging environment.

we highlight here shows a drone flying straight down a long hallway-like interior of the plane (as seen in Fig. 5.3). We chose this environment and specific test-case to highlight a degeneracy case for systems that use only ORB features that mis-match in situations with a high number of near-identical points in a repetitive texture. In the tests conducted, this environment was shown to be extremely challenging for ORB-SLAM2, which drifts by 1.8 meters over the course of a 12 meter flight along the Y-axis (forward). The ground truth flight path is shown in Fig. 5.8 with a comparison to the drone positions from ORB-SLAM2 and our Multi-Level Semantic SLAM.

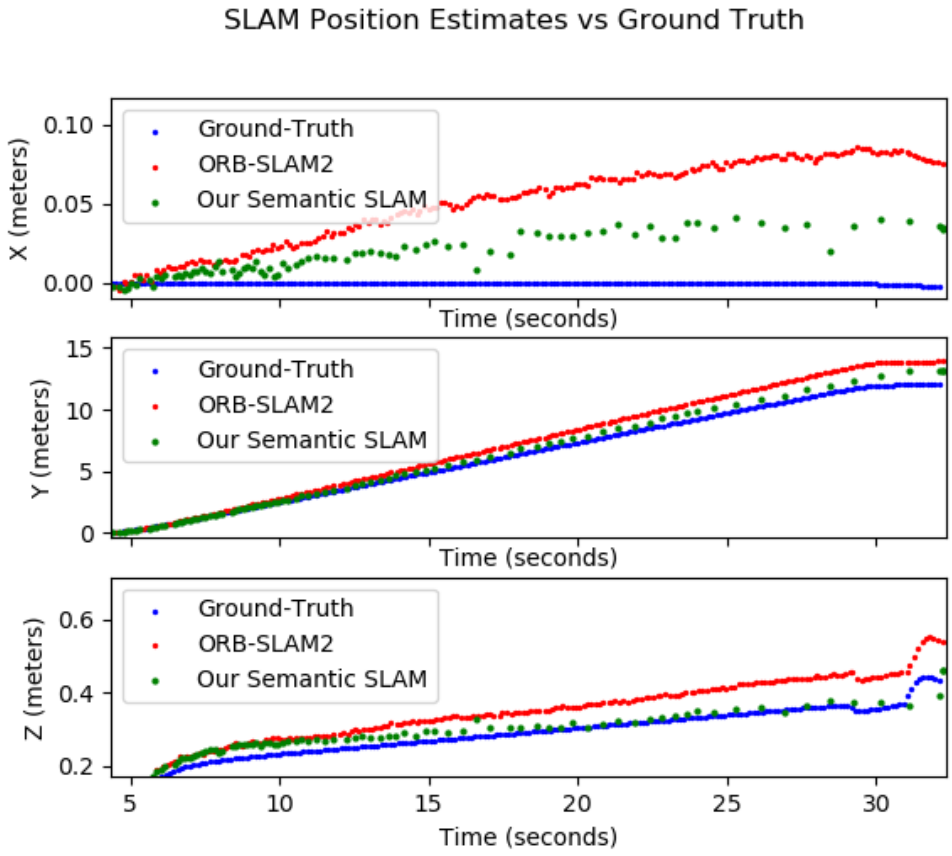

Figure 5.8: Position estimates from ORB SLAM2 and the Multi-Level Semantic SLAM vs. Ground Truth

Fig. 5.9 shows a comparison of the error along the flight path between ground

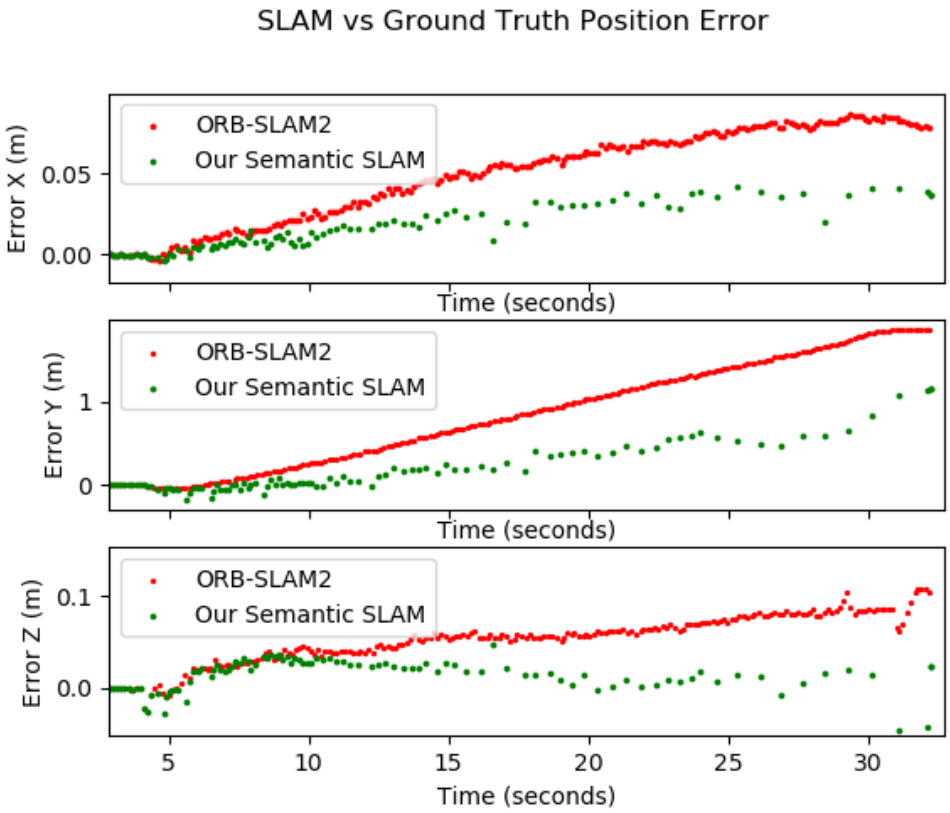

Figure 5.9: Tracking errors from ORB SLAM2 and the Multi-Level Semantic SLAM vs Ground Truth

truth and both ORB-SLAM2 and our Multi-Level Semantic SLAM. The rate of error accumulation of ORB-SLAM2 along the Y-axis appears relatively consistent along the flight path as the floor textures repeat along the entire flight. It is noticed that the rate of error from our system starts off matching that of the ORB-SLAM2, but as semantic objects come into view our system starts to accumulate error at a much lower rate until the end of the flight where few objects are left in view and the error begins to accumulate faster as it must again focus on the ORB features that are still in view. Over the course of the full 12m flight, the ORB-SLAM2's estimate of drone position drifted 187cm, while the estimate by our method drifted only 115cm, a difference of 40% improvement. At its best point, while semantic

objects are still within view around the 26 second mark, ORB-SLAM2's position estimate has an error of 152cm, while the position estimate by our system has an error of only 47cm, for an improvement of 70% in tracking accuracy.

We also tested our system on the simulated exterior of an airplane. See Fig. 5.1. As the drone moves forward, the ORB SLAM2 failed after 5 seconds due to severe mismatches of ORB features. This shows that even with our system, there are still limitations when the environment reaches a point of extreme cases of sparse and repetitive features. Future works could explore adding IMU information to our method to improve further for edge cases such as the above exterior environment.

We also tested our ML Semantic SLAM method on a more standardized environment of a bathroom with a variety of features including repetitive flooring tiles. The repeating features pose an additional challenge over typical environments. Our ML Semantic SLAM method showed modest gains of a 2% reduction in error over the course of the trajectory which completed three movements of 1.95 meters in and out of the bathroom and then returned to the same starting position. Full-trajectory ground-truth data was not available given that this data was collected in the real-world without a motion capture system available. However, the position was measured relative a fixed reference point at the start and end of the trajectory. The output of our Semantic SLAM map of the bathroom environment is seen in Fig. 5.10.

Both of the above datasets (Simulated Airplane and Real-World Bathroom) are open-sourced. They are available online at [15].

Figure 5.10: The semantic features and map from our SLAM is overlayed on the RGB input from this semi-sparse bathroom environment with repetitive floor tiles

Chapter 6

Discussion and Future Work

This chapter provides discussion of results and future extensions of our approach.

6.1 Summary of Results

In this thesis, we have presented the development and evaluation of our Multi-Level Semantic SLAM (ML Semantic SLAM) method, demonstrating its effectiveness in various simulation environments and real-world scenarios. Our method capitalizes on various low-level and high-level semantic features to reduce tracking errors in sparse environments and create a more semantically rich map. Our approach improves the overall performance of SLAM systems by incorporating multiple feature types and levels, including planes, lines, semantic regions, and objects, to address the sparsity of environments with limited distinguishable keypoint ORB features.

As demonstrated in the above experimental results section, our ML Semantic SLAM system has achieved significant error reductions, with up to 70% improvement in an extremely sparse simulated environment characterized by very few distinct features, and a modest 2% error reduction in a less-sparse real-world bathroom environment. These performance improvements can be attributed to the effective

integration of diverse feature types at various semantic levels. The success of our method in an extremely challenging environment highlights its potential to offer similar improvements in other repetitive and sparse environments, such as long hallways devoid of distinctive floor or wall textures.

6.2 Future Extensions to ML Semantic SLAM

Despite the promising results, there are still areas for improvement and expansion in future works. The following are some possible directions to further enhance the performance and capabilities of our ML Semantic SLAM method:

- **Incorporate more sensor modalities:**

  The current implementation of our ML Semantic SLAM system primarily utilizes an RGB-D camera. Integrating additional sensors, such as IMUs or additional cameras could help improve the accuracy and robustness of the system in more challenging environments or edge cases.

- **Expand object detection and semantic segmentation capabilities:**

  Currently, our method leverages a predefined set of semantic objects. Extending the variety and types of detectable objects would enable our system to work more effectively in diverse environments and cater to specific application domains.

- **Develop an adaptive and dynamic weighting strategy:**

  Our current approach uses a fixed weighting strategy for combining semantic and ORB features (described in the bundle adjustment section). Future work could explore a more adaptive and dynamic weighting strategy that adjusts according to the environment, object confidence, feature similarity scores, and

other factors, further improving the system's overall performance.

- **Investigate real-time performance optimization:**

  The current implementation of our ML Semantic SLAM method may not be optimal for real-time performance, as it runs slower than 10FPS. Future work could explore methods to optimize the system for real-time applications, such as reducing computational complexity, accelerating object detection and segmentation, or leveraging hardware acceleration techniques.

- **Enhance the realism of simulation environments:**

  Our experimental evaluation mainly focused on simulated environments (other than one real-world scene). We explored a few methods to enhance the environments, such as augmenting with real-world textures and reconstructing from public videos with NeRFs, but we didn't explicitly test on these yet. Future work could continue to explore and test on more realistic simulation environments, such as those based on Neural Radiance Fields (NeRFs) or photogrammetry, to better understand and evaluate the performance of our ML Semantic SLAM system in real-world situations.

In conclusion, the Multi-Level Semantic SLAM method shows great promise in enhancing the performance of traditional SLAM systems in various environments. As we continue to refine and expand upon our system, we hope to further advance the field of robotics and enable more reliable and accurate SLAM for a wide range of applications.

6.3 Proposed Extension: Active Semantic SLAM

While we detailed specific future work directions for the ML Semantic SLAM specifically, we also have a detailed plan of how the work could be extended to an Active SLAM method.

6.3.1 Context

Active SLAM aims to enhance the capabilities of traditional SLAM techniques as it actively utilizes the generated map to enable a robot to explore and navigate its environment autonomously. Existing active SLAM methods have mostly focused on open frontier exploration for geometric coverage or object discovery. One research area that is still unexplored is the application of Active SLAM to a semantic exploration task. In an application such as airplane inspection, there is a desire to determine semantic labels for areas of interest in the airplane, such as windows for inspection. This is challenging as off-the-shelf semantic segmentation networks frequently face difficulties in predicting correct and consistent class labels when observing objects at varying distances. To address these challenges, we propose a research direction centered on Active Semantic SLAM for confident semantic labeling and exploration.

6.3.2 Research Objectives

The primary objectives of this research are to develop an Active Semantic SLAM approach capable of:

1. Exploring unknown areas with the goal of achieving confident semantic labels for goal semantic classes (e.g., windows for inspection).

2. Improving and filtering noisy labels from off-the-shelf semantic segmentation networks.

3. Investigating multi-agent coordination for semantic coverage with different sensors.

6.3.3 Active SLAM for Semantic Coverage

This research will focus on developing an active SLAM approach that ensures semantic coverage of semantically-important areas without exhaustive coverage of the entire environment. This could leverage modified coverage planning methods for active SLAM with an added semantic component and employing learning-based techniques to predict exploration directions.

Additional methods to be explored include Neural Radiance Fields (NeRF) for predicting the location of semantics of interest in the environment and incorporating a multi-resolution mapping component for efficient data storage and representation.

6.3.4 Adaptive Ideal Distance for Semantic Labeling

To enhance the quality of semantic labeling, we propose to develop an adaptive distance threshold for determining the confidence of the semantic prediction based on the distance to the observed pixel. This approach would define an ideal distance for each class and adjust it over time based on the size of the class regions and the distribution of confidence scores in the regions. By learning the ideal distance for each class and adjusting the confidence accordingly, the proposed system can achieve confident semantic labels without relying on prior knowledge or training on a specific dataset.

6.3.5 Multi-resolution Mapping

Another essential aspect of this research is the investigation of multi-resolution mapping techniques for reducing storage space and focusing computation on regions of interest that require higher resolution. By employing techniques such as OctoMaps with varying resolutions, the proposed system can achieve increased resolution for specific regions of the map as more observations are collected, while maintaining lower-resolution maps for uninteresting areas.

6.3.6 Remarks

The proposed research on Active Semantic SLAM for confident semantic labeling aims to address the limitations of traditional active SLAM methods and enhance the quality of semantic labeling in various environments. By focusing on the development of an active SLAM approach that achieves confident semantic labels, improves noisy labels from existing segmentation networks, and investigates multi-agent coordination for semantic coverage, this research has great potential for inspection applications and beyond.

6.4 Conclusions

In this thesis, we have presented a comprehensive exploration of the development, implementation, and evaluation of our Multi-Level Semantic SLAM method, which addresses the challenges of accurate and reliable localization and mapping in sparse and repetitive environments. Our approach stands out by leveraging both low-level and high-level semantic features to create semantically-detailed maps and achieve significant error reduction.

We provided an in-depth analysis of how our method works, with step-by-step

details of feature extraction and the bundle adjustment formulation that ties all the features together. We showcased the robustness and versatility of our approach in various simulated and real-world environments, emphasizing its potential for improved performance in other challenging scenarios, such as airplane environments for inspection.

Additionally, we proposed a research direction focused on Active Semantic SLAM for Semantic Coverage Exploration, which aims to expand traditional Active SLAM techniques and enhance the quality of semantic maps in new environments. This research proposal underlines the importance of continual innovation and expansion in the field, and we believe it provides a solid foundation for future investigations.

As a final remark, this thesis not only contributes to the advancement of SLAM systems and robotics but also serves as a catalyst for future research and development in the area. We hope that our findings inspire further exploration, refinement, and implementation of novel approaches to SLAM, leading to more accurate and reliable systems that benefit a broad range of real-world applications.

Bibliography

[1] Agarwal, S., Mierle, K., and Team, T. C. S. Ceres Solver, 3 2022.

[2] Burkard, R., Dell'Amico, M., and Martello, S. *Assignment problems: revised reprint*. SIAM, 2012.

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

[4] Chandraker, M., Lim, J., and Kriegman, D. Moving in stereo: Efficient structure and motion using lines. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 1741–1748.

[5] Douglas, D. H., and Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization 10*, 2 (1973), 112–122.

[6] Epic Games. Unreal engine.

[7] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Springer International Publishing, Cham, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[8] Goldberg, A. V., and Kennedy, R. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming 71*, 2 (1995), 153–177.

[9] Guan, P., Cao, Z., Chen, E., Liang, S., Tan, M., and Yu, J. A real-time semantic visual slam approach with points and objects. *International Journal of Advanced Robotic Systems 17*, 1 (2020), 1729881420905443.

[10] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research 31*, 2 (2012), 216–235.

[11] Kennedy Jr, J. R. *Solving unweighted and weighted bipartite matching problems in theory and practice*. Stanford University, 1995.

[12] Kim, U.-H., Kim, S.-H., and Kim, J.-H. Simvodis: Simultaneous visual odometry, object detection, and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 1 (2020), 428–441.

[13] Klein, G., and Murray, D. Improving the agility of keyframe-based slam. In *European conference on computer vision* (2008), Springer, pp. 802–815.

[14] Koenig, N., and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (2004), vol. 3, IEEE, pp. 2149–2154.

[15] Krueger, K. Sparse slam datasets, 2023.

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

[17] Labbé, M., and Michaud, F. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics 36*, 2 (2019), 416–446.

[18] Leonard, J., and Durrant-Whyte, H. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91* (1991), pp. 1442–1447 vol.3.

[19] Lianos, K.-N., Schonberger, J. L., Pollefeys, M., and Sattler, T. Vso: Visual semantic odometry. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 234–250.

[20] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[21] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM 65*, 1 (2021), 99–106.

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

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

[24] Perron, L., and Furnon, V. Or-tools.

[25] Qian, Z., Fu, J., and Xiao, J. Towards accurate loop closure detection in semantic slam with 3d semantic covisibility graphs. *IEEE Robotics and Automation Letters 7*, 2 (2022), 2455–2462.

[26] Qian, Z., Patath, K., Fu, J., and Xiao, J. Semantic slam with autonomous object-level data association. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), IEEE, pp. 11203–11209.

[27] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[28] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision* (2011), Ieee, pp. 2564–2571.

[29] Rusu, R. B., and Cousins, S. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, May 9-13 2011).

[30] Shah, S., Dey, D., Lovett, C., and Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics* (2017).

[31] Sklansky, J. Finding the convex hull of a simple polygon. *Pattern Recognition Letters 1*, 2 (1982), 79–83.

[32] Suzuki, S., et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing 30*, 1 (1985), 32–46.

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

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

[35] Zhang, G., and Suh, I. H. A vertical and floor line-based monocular slam system for corridor environments. *International Journal of Control, Automation and Systems 10*, 3 (2012), 547–557.

[36] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[37] Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision* (2018).