Navigation and Dense Semantic Mapping with Autonomous Robots for Environmental Monitoring

by

Lakshmi Gana Prasad Antervedi

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved November 2021 by the
Graduate Supervisory Committee:

Jnaneshwar Das, Chair
Roberta Martin
Hamid Marvi

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Autonomous Robots have a tremendous potential to assist humans in environmental monitoring tasks. In order to generate meaningful data for humans to analyze, the robots need to collect accurate data and develop reliable representation of the environment. This is achieved by employing scalable and robust navigation and mapping algorithms that facilitate acquiring and understanding data collected from the array of on-board sensors. To this end, this thesis presents navigation and mapping algorithms for autonomous robots that can enable robot navigation in complex environments and develop real time semantic map of the environment respectively. The first part of the thesis presents a novel navigation algorithm for an autonomous underwater vehicle that can maintain a fixed distance from the coral terrain while following a human diver. Following a human diver ensures that the robot would visit all important sites in the coral reef while maintaining a constant distance from the terrain reduces heterscedasticity in the measurements. This algorithm was tested on three different synthetic terrains including a real model of a coral reef in Hawaii. The second part of the thesis presents a dense semantic surfel mapping technique based on top of a popular surfel mapping algorithm that can generate meaningful maps in real time. A semantic mask from a depth aligned RGB-D camera was used to assign labels to the surfels which were then probabilistically updated with multiple measurements. The mapping algorithm was tested with simulated data from an RGB-D camera and the results were analyzed.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor and committee chair Dr. Jnaneshwar Das for his support and guidance throughout the course of my Master's program. Thank you for making me a part of exciting projects and facilitating an environment that is conducive to innovation and critical thinking.

I would like to express my gratitude to Dr. Roberta Martin for providing invaluable insight and direction on the scientific aspects of the coral mapping problem. I would also like to thank Dr. Hamid Marvi for helping me develop a solid background in the theoretical concepts of robotics that I have applied throughout my research.

Last but not the least, I would like to thank my fellow DREAMS lab members Rakshith Vishwanath, Zhiang Chen, Harish Anand, Devin Keating, Dr. Luiza Aparecido, Sarah Bearman, and Alex Goldman for the stimulating discussions during our lab meetings.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Thesis Statement

Efficient and real time navigation and semantic mapping techniques for autonomous robots can enable safe and fast mapping and monitoring of environments.

1.2 Overview

This thesis presents navigation and semantic mapping algorithms for autonomous robots that can assist humans in data acquisition and analysis. A real time semantic mapping algorithm is presented that can develop meaningful maps from the on-board RGB-D camera. Chapter 2 presents background and my experiments with Terrain Relative Navigation on quadrotor drone in a simulated environment. Based on the lessons learnt from the Terrain Relative Navigation on a drone, I developed Terrain Relative Diver Following algorithm for an autonomous underwater vehicle that can follow a human diver while maintaining a fixed distance from the terrain. This work was presented in the 17th IEEE Conference on Automation Science and Engineering (IEEE CASE) 2021 and the paper has been included in full in chapter 3. Chapter 4 presents the dense semantic surfel mapping technique that uses semantic masks obtained from a segmentation pipeline to encode semantic information to the surfels that are probabilistically updated with new measurements. The algorithm is tested with simulated data and the results are presented. 5 presents a summary of the work presented and directions for future work.

1.3 Contributions

This thesis presents the following contributions:

- Terrain Relative Diver Following algorithm for an autonomous underwater vehicle that enables the robot to follow a human diver while maintaining a fixed distance from the terrain.
- Extensive analysis of the algorithm on three different terrains including a real model of a coral reef.
- A Real Time Dense Semantic Surfel Mapping technique that assigns and updates semantic information to surfels using segmentation masks obtained from color images.
- The code for the navigation algorithm presented has been made available to the research community.

Chapter 2

TERRAIN RELATIVE NAVIGATION

2.1 Introduction

Terrain Relative Navigation (TRN) is a navigation approach historically used by sub-marines and missiles prior to the advent of Global Positioning System (GPS) to navigate to the target locations. A recent implementation of TRN approach was used by perseverance rover to land on the target site in mars.

Another popular navigation approach that uses the Terrain information for path

Figure 2.1: Artist's Depiction of the Mars 2020 Perseverance Rover Landing on Mars

Source: https://science.nasa.gov/technology/technology-highlights/terrain-relative-navigation-landing-between-the-hazards

planning is when a robot maintains constant distance from the terrain. For some environmental monitoring missions, this navigation approach is critical as the terrain varies tens of meters in altitude. As the depth cameras have a limited range, it becomes necessary to vary altitude of the sensor in order to obtain usable data for mapping and monitoring applications. This approach also has the advantage of increasing the consistency of the imagery as the mapping payload stays at approximately constant distance from the terrain which consequently reduces the heteroscedasticity in the noise of the measurements. In this thesis, I implemented a Terrain Following approach that obtains terrain information from depth camera and uses that to plan and track a trajectory to maintain a fixed distance from the terrain. This approach was tested in Gazebo Simulation Environment (Koenig and Howard (2004)). A 3D model of a real fault scarp in Bishop, California was used to provide a realistic setting to test the Terrain Following algorithm.

2.2 Non-Linear Model Predictive Control for Trajectory Tracking

A non linear Model Predictive Control was used for tracking trajectory that maintained a constant distance from the terrain. I used the Non-linear MPC implementation presented in Kamel *et al.* (2016) in my experiments for testing Terrain Following on realistic terrains. The non-linear MPC problem can be formulated as an optimal control problem as shown in equation 2.1

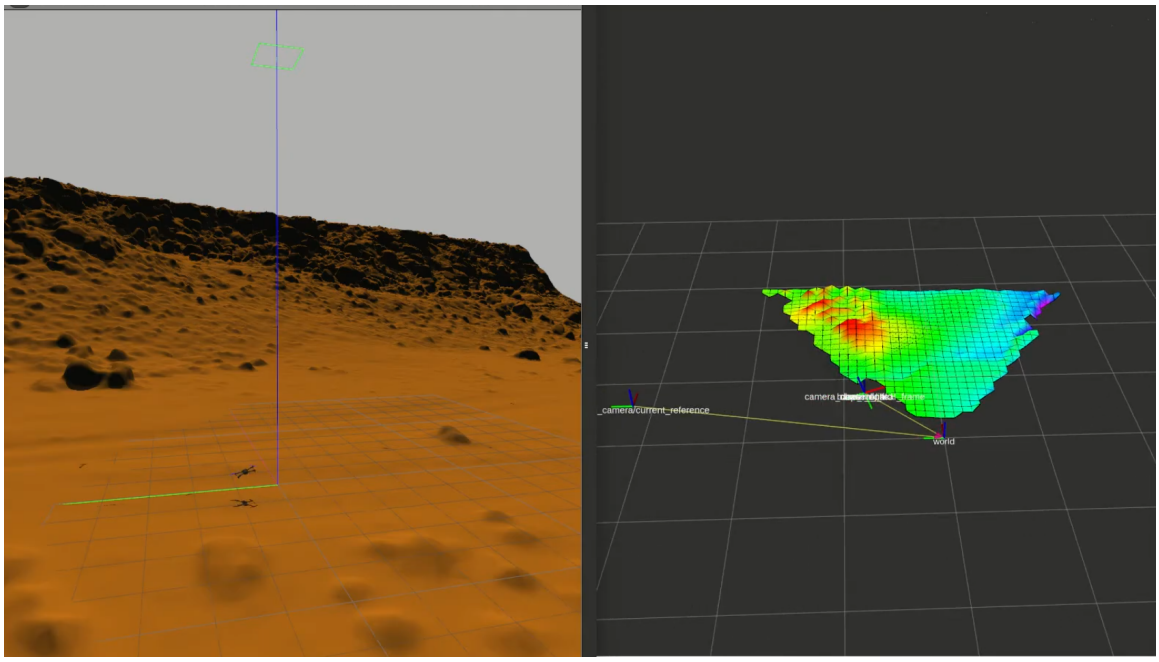

Figure 2.2: Terrain Relative Navigation Using Iris Quadrotor in the Simulated Bishop Terrain

$$
\begin{aligned}
&\min_{\boldsymbol{U},\boldsymbol{X}} \int_{t=0}^{T} \|\boldsymbol{x}(t) - \boldsymbol{x}_{ref}(t)\|^2_{\boldsymbol{Q}_x} + \|\boldsymbol{u}(t) - \boldsymbol{u}_{ref}(t)\|^2_{\boldsymbol{R}_u}\, dt \\
&+\|\boldsymbol{x}(T) - \boldsymbol{x}_{ref}(T))\,\|^2_{\boldsymbol{P}} \\
&\text{subject to} \\
&\quad \dot{\boldsymbol{v}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\
&\qquad \boldsymbol{u}(t) \in \mathbb{U} \\
&\qquad \boldsymbol{x}(0) = \boldsymbol{x}\,(t_0)
\end{aligned}
\tag{2.1}
$$

where $x = (p^T v^T\ \phi\ \theta\ \psi)^T$ represents the state of the robot where p denotes the position of the robot, v denotes the velocity, ϕ, θ and ψ denote the roll, pitch and yaw angles of the robot. u is the control vector represented as $(\phi_{cmd}\ \theta_{cmd}\ T_{cmd})^T$ where ϕ_{cmd}, θ_{cmd} and T_{cmd} are the commanded roll, pitch and thrust values.

The function f denotes the dynamics of the aerial drone and is given by the equations 2.2

$$
\begin{aligned}
\dot{p} &= v, \\
\dot{v} &= \frac{1}{m}\left(\boldsymbol{R}_{IB}\sum_{i=0}^{N_r}\boldsymbol{F}_{T,i} - \boldsymbol{R}_{IB}\sum_{i=0}^{N_r}\boldsymbol{F}_{aero,i} + \boldsymbol{F}_{ext}\right) \\
&\quad + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \\
\dot{R}_{IB} &= \boldsymbol{R}_{IB}\lfloor\boldsymbol{\omega}\times\rfloor \\
\boldsymbol{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega}\times\boldsymbol{J}\boldsymbol{\omega} + \mathcal{A}\begin{bmatrix} n_1^2 \\ \vdots \\ n_{N_r}^2 \end{bmatrix}
\end{aligned}
\tag{2.2}
$$

where m is the mass of the drone, $\boldsymbol{R}_{IB} \in \mathbb{R}^{3x3}$ is the rotation matrix representing orientation of the body frame of the robot in an inertial frame. $F_{aero,i}$ is the aerodynamic forces acting on rotor i, F_{ext} is the external force acting on the robot, g is the acceleration due to gravity, ω represents the angular velocity of the robot in the body frame, $\boldsymbol{J}$ is the inertia matrix of the robot, $\mathcal{A}$ is the actuation matrix to convert rotor velocities given by n_i to the corresponding force and torque.

The solution to this optimal control problem is obtained using a real time iteration scheme based on Gauss-Newton to approximate the optimization problem. This is done by a solver generated by the ACADO toolkit (Houska *et al.* (2011)).

2.3 Terrain Mapping

In order to maintain a fixed distance from the terrain, the robot needs to have a decent estimate of the terrain at all times. There are multiple approaches to storing and accessing the information such as using raw point cloud data, using a range sensor (e.g. ultrasonic sensors) etc. However using raw data directly can lead to

rough trajectories as the sensor data is usually noisy and needs some kind of pre-processing. Fankhauser *et al.* (2014) presented a method for terrain mapping for their quadruped robot that uses raw point cloud data as input to generate real time robot centric elevation map. I used this method to generate a representation of the terrain that is passed as input to the trajectory optimization method described in section 2.2.

The method for terrain mapping as presented by Fankhauser *et al.* (2014) involves building an elevation grid map where each cell in the grid stores the height estimate $\hat{h}$ and variance σ_h^2. The advantage of their method is that the elevation map being built is robot-centric, and not tied to any inertial frame. Hence the drift acculmulated due to errors in the pose estimation do not translate to the elevation map.

The height measurement (p, σ_p^2) is fused with the existing elevation map estimation $(\hat{h}, \sigma_h^2)$ using the following update equations

$$\hat{h}^+ = \frac{\sigma_p^2 \hat{h}^- + \hat{\sigma}_h^{2-} \tilde{p}}{\sigma_p^2 + \hat{\sigma}_h^{2-}}, \quad \hat{\sigma}_h^{2+} = \frac{\hat{\sigma}_h^{2-} \sigma_p^2}{\hat{\sigma}_h^{2-} + \sigma_p^2} \tag{2.3}$$

where previous estimates are denoted by $-$ sign and the new estimates are denoted by the $+$ sign.

Using this approach, the quadrotor drone was able to climb the simulated Bishop Terrain that is approximately 20m in height. The video showing the terrain following and the climb is available here.

2.4 Terrain Relative Diver Following with Autonomous Underwater Vehicle

Leveraging the lessons learnt in implementing a terrain following system for an aerial robot, I developed an algorithm for an underwater vehicle that can follow a human diver while maintaining a fixed distance from the coral reef terrain. This work was submitted to the 17th IEEE Conference on Automation Science and Engineering

Conference in August 2021. My contributions to the paper were (i) writing the diver following algorithm that used position estimates from the diver detection module to predict pose of the diver, (ii) training YOLOV3 on custom dataset to detect diver, (iii) developing terrains for simulation experiments in Gazebo and (iv) performing the experiments in Gazebo on three different terrains. This paper has been presented in full in chapter 3.

Chapter 3

TERRAIN RELATIVE DIVER FOLLOWING WITH AUTONOMOUS UNDERWATER VEHICLE FOR CORAL REEF MONITORING

The work on Terrain Relative Diver Following was published in 17th IEEE Conference on Automation Science and Engineering, 2021. Terrain-Relative Diver Following with Autonomous Underwater Vehicle for Coral Reef Mapping, 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 08/2021. DOI: 10.1109/CASE49439.2021.9551624

3.1 Abstract

Coral reef mapping is an indispensable step in coral conservation efforts across the globe. Monitoring reefs at regular intervals helps conservationists understand and address the problems causing coral reef degradation. Autonomous Underwater Vehicles (AUVs) have a tremendous potential to assist humans in these efforts. Delegating mapping and measurement acquisition tasks to AUVs would not only limit the number of human divers required for the missions but could also improve the quality of the maps developed. Consistency in imagery and spectroscopic measurements could be significantly improved by keeping the imagery payload at a fixed distance from the reefs to reduce heteroscedasticity in the measurements. To this end, I present a *Terrain-Relative Diver Following* system for an AUV that can follow a human diver while maintaining a fixed distance from the terrain. The proposed system consists of separate modules for diver detection, tracking, and terrain following. I extensively

tested the system in Gazebo simulation environment with three different terrain models, including a terrain model of a coral reef in Honaunau Bay, Hawaii. To the best of my knowledge, this is the first diver following system that also carries out terrain-relative navigation, ensuring minimal variation of distance to the terrain. I have released the code for the system, and the datasets used in the detection module.

3.2 Introduction

Underwater Vehicles are deployed in many underwater exploration and inspection tasks such as ship hull inspection (Vaganay *et al.* (2006)), monitoring and repairing underwater infrastructure (Petillot *et al.* (2002)), etc. Recently, there has been an increased interest in developing low-cost micro AUVs ($< 1m$ in length) that could operate completely autonomously underwater (Hackbarth *et al.* (2015), Edge *et al.* (2020)). Also, there has been a push to develop interactive capabilities in such systems that can assist humans in tasks that are dirty, dull, and dangerous.

I consider the application of coral reef monitoring where an AUV equipped with the necessary instruments can assist a human diver in collecting measurements, mapping, and surveying the reef. The quality of measurements can be significantly improved by keeping the cameras at a constant distance from the reef throughout the mission. Besides reducing the heteroscedasticity in the measurements, it would also reduce the cognitive load on the diver who is already in a hostile underwater environment.

A typical diver-AUV mission for coral reef monitoring would include a diver guiding the robot to important sites around the reef and the robot gathering the data while maintaining a fixed distance from the coral terrain.

A major challenge in developing such an algorithm stems from the often conflicting nature of the two mission objectives: *Diver Following* and *Terrain Following*. For

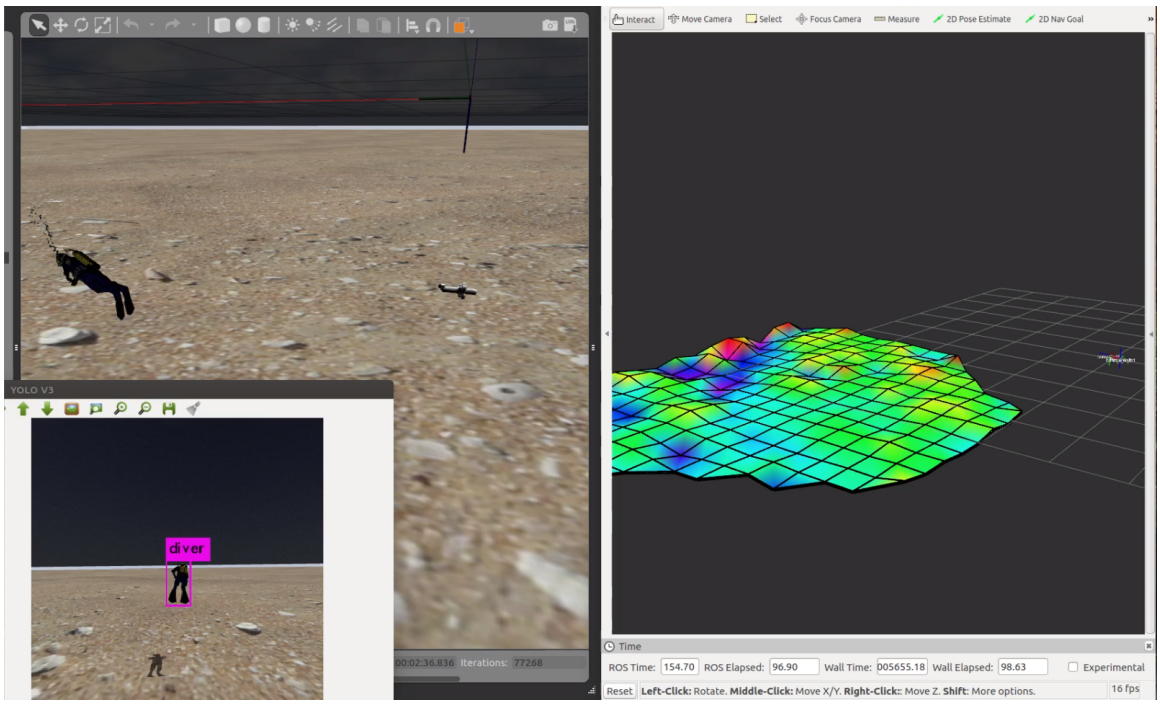

Figure 3.1: Detecting (Bottom Left) and Tracking the Diver While Generating Elevation Map (Right) To Maintain a Fixed Distance From the Terrain.

instance, the nature of the terrain often requires the robot to change elevation which might lead to the diver moving out of the field of view of the robot's camera. I used a modular approach for the algorithm that can decouple and satisfy the two objectives of the target trajectory.

In sum, the following are the contributions of this paper:

- a navigation algorithm that enables an AUV to follow a diver while maintaining a constant distance from the submarine terrain.
- extensive testing of the algorithm on multiple terrains: (i) Flat, (ii) Sinusoidal, and (iii) Coral Reef terrain models.
- I have released the the code for the navigation algorithm and the model for diver detection used in this paper. The code and the material are available at

github.com/algprasad/diver_follower

3.3 Related Work

The work presented in this document lies at the intersection of two sub-fields of robotics: *Person Following* (Gockley *et al.* (2007), Islam *et al.* (2019), Müller *et al.* (2008)) and *Terrain Relative Navigation* (TRN) (Johnson and Montgomery (2008), Krukowski and Rock (2016), Meduna *et al.* (2010)). *Person Following* falls under the broader field of *Human-Robot Interaction* and is a crucial capability for mobile robots deployed in the domains of manufacturing, healthcare, etc. Islam *et al.* (2019) presented a comprehensive survey of the existing *Person Following* algorithms in the operational domains of ground, air and underwater.

A recent survey on TRN approaches for AUV navigation is presented in Melo and Matos (2017). Most of the literature on TRN for AUVs is addressed for large AUVs that are designed for long-term and long-range missions (Meduna *et al.* (2008)). Consequently, a majority of the current algorithms use sophisticated range measurement sensors that cannot be used for small AUVs.

My approach to TRN uses the elevation mapping technique presented by Fankhauser *et al.* (2014) that employs a point cloud to generate an elevation map in real time. I used the open sourced implementation from Fankhauser *et al.* (2018) in this work for generating an elevation map with a stereo camera mounted underneath the AUV.

The past decade has seen an increased interest in developing micro Autonomous Underwater Vehicles thanks to the proliferation of Micro Aerial Vehicles and the frameworks (Meier *et al.* (2015)) supporting their development. Hackbarth *et al.* (2015) presented a quadrotor-design based, low-cost autonomous underwater platform called *HippoCampus*, which I use for the experiments in this paper. Edge *et al.* (2020) designed *LoCo AUV*, a general-purpose vision guided AUV with human-interaction

capabilities in the form of *diver following* and *gestural control*.

The problem of following a diver with an underwater vehicle has been addressed by Islam *et al.* (2019) where the authors used deep neural networks to detect a diver. They trained their model on images of divers that were obtained in real underwater settings. The focus of the paper is not to present a better diver detection algorithm, but to present a complete algorithm that can detect, track, and follow a diver while also following the terrain. To that end, I used a 3D model of a scuba diver obtained from Nsfr750 and CGTrader (2014) and trained the diver detection module on the synthetic diver images of the 3D model.

3.4 Problem Formulation

The goal of the *Terrain-Relative Diver Following* algorithm is to generate and track a trajectory that not only follows a diver but also maintains a constant distance from the terrain. The robot is equipped with two stereo cameras, one pointing forward to detect and track the diver and the other pointing downward for generating elevation maps. Also, I assume that the robot has access to it's pose estimates.

The robot should be able to detect the diver in the image frames, estimate and track the diver's position throughout the mission. The robot should also be able to incorporate terrain information in the trajectory.

3.5 System Description

Fig. 3.2 depicts the pipeline for *Terrain-Relative Diver Following* algorithm. Each module in the pipeline is described in the following subsections.

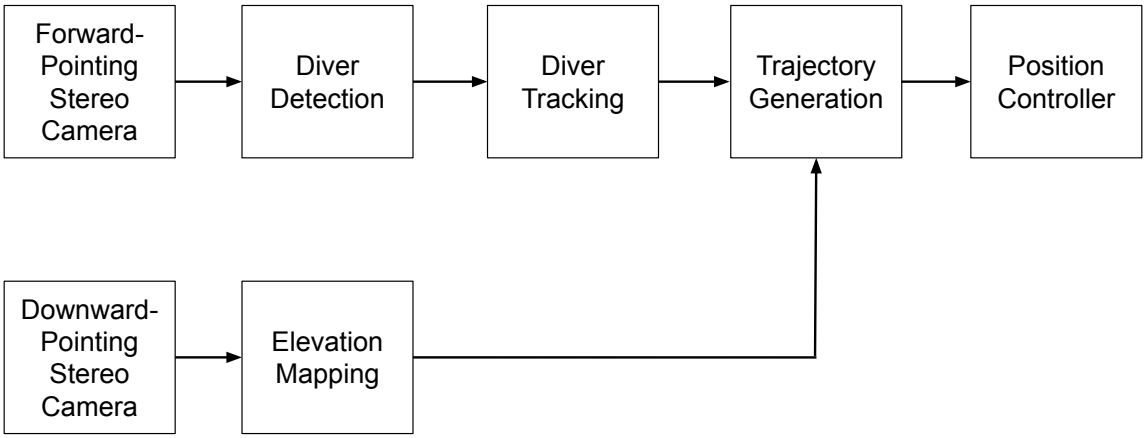

Figure 3.2: Pipeline for the *Terrain-Relative Diver Following* Algorithm.

3.5.1 Diver Detection

With the objective to make the simulations in Gazebo (Koenig and Howard (2004)) as close to real conditions as possible, I used a 3D model of a scuba diver obtained from Nsfr750 and CGTrader (2014). I re-trained a YOLO v3 by Redmon *et al.* (2015) model on 300 images of the diver at different orientations. I used the ROS (Stanford Artificial Intelligence Laboratory et al. (2018)) based implementation of the YOLO v3 model from Bjelonic (2018) and obtained a detection rate of 20 frames per second with NVIDIA GTX 1050Ti GPU. Fig. 3.3 shows the diver 3D model and a successful diver detection with YOLO v3.

3.5.2 Diver Tracking

The diver tracking system consists of a calibrated forward facing stereo camera that is used to obtain a depth image and the corresponding point cloud. The bounding box from the detection module is used to determine the points that correspond to the diver. I sample the points that lie along the orthogonal lines passing through the centre of the bounding box. Formally, the position of the diver in the global reference

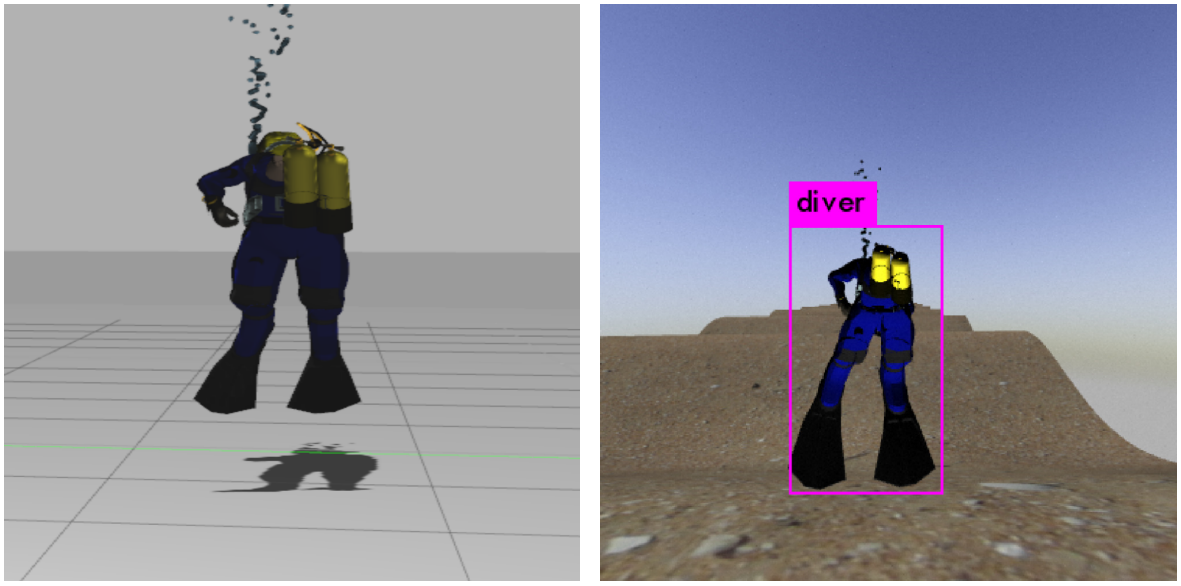

Figure 3.3: Left Figure Shows the 3D Model of a Diver Used in the Experiments. Right Figure Shows the Diver Detection From the Front-Facing Camera Using YOLO v3 in Gazebo.

frame is given as

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{3.1}$$

where N is the number of points in the set X. The set X consists of global positions of all the points in the point cloud that have the corresponding pixel coordinates (u, v) satisfying the condition $u = (x_{min} + x_{max})/2$ or $v = (y_{min} + y_{max})/2$, where $x_{min}, x_{max}, y_{min}$ and y_{max} are the coordinates of the bounding box obtained from the diver detection step.

The position of the diver so obtained is then fed into a Kalman filter (Kalman (1960)) that outputs a filtered position of the diver. The purpose of the Kalman filter is two-fold: firstly, it filters the noisy measurements of the diver's position obtained from the raw point cloud, and secondly, it maintains a position of the diver even

when the diver is out of the view of the camera. The latter happens quite often with the *HippoCampus* AUV as it is under-actuated and has to change the pitch angle to change elevation. This moves the camera as well, thereby making the diver move out of the robot's camera frame.

I assume that the diver moves with constant velocity between consecutive steps of the filter. The *predict* step of the Kalman filter is formulated as follows

$$\begin{aligned} \tilde{X}_{k+1} &= AX_k \\ \tilde{P}_{k+1} &= AP_kA^T + Q \end{aligned} \tag{3.2}$$

where $X_k = [x_k, y_k, z_k, \dot{x_k}, \dot{y_k}, \dot{z_k}]$ is the vector of the global position and velocity of the diver, and $A = \begin{bmatrix} 1 & 0 & 0 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ where $\delta t = 0.05$ is the time step and depends on the frequency of the filter. We initialize P_0 as a zero matrix and take $Q = diagonal(\epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon)$, where $\epsilon = 1e - 13$.

The measurement update steps are given as

$$\begin{aligned} \tilde{z} &= \bar{z} - H\tilde{X}_{k+1} \\ Z &= H\tilde{P}_{k+1}H^T + R \\ K &= \tilde{P}_{k+1}H^TZ^{-1} \\ X_{k+1} &= \tilde{X}_{k+1} + K\tilde{z} \\ P_{k+1} &= \tilde{P}_{k+1} - KZ\tilde{P}_{k+1} \end{aligned} \tag{3.3}$$

where $\bar{z}$ is the vector containing the measured values of the position of the diver

obtained from the point cloud and H is given as $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$. R is the measurement noise matrix which I estimated to be equal to $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Figure 3.4 shows that the filter is able to *smooth* the noisy measurements (shown in blue) and output the filtered values of the diver's position (shown in red).

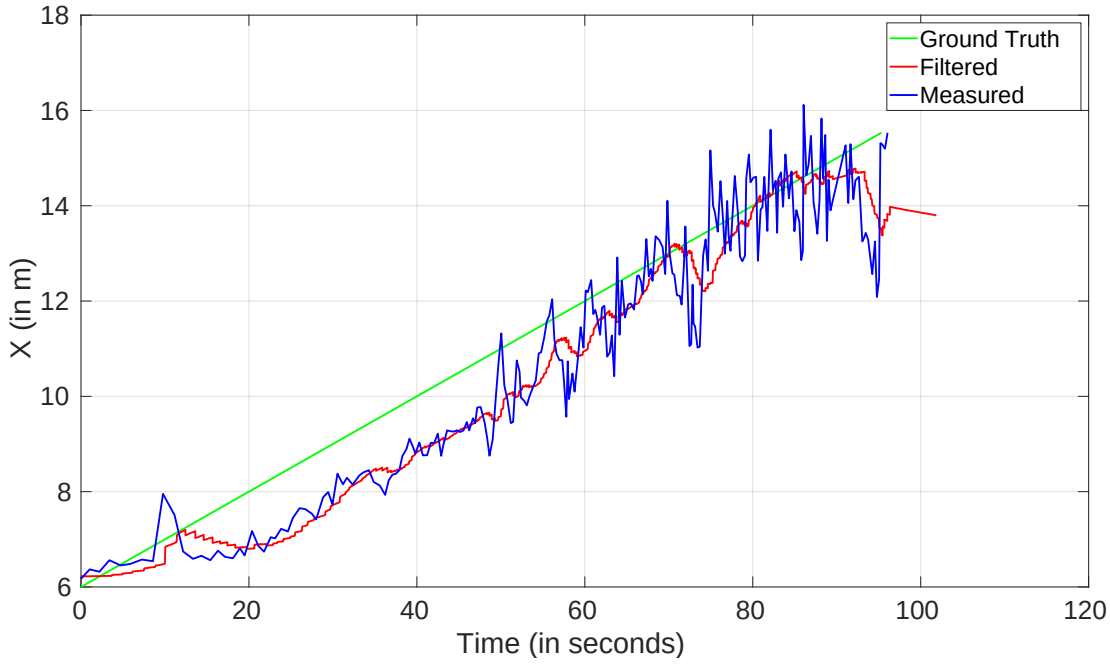

Figure 3.4: Performance of the Diver Tracking Filter (Red) Compared Against Ground Truth (Green) and Noisy Measurements (Blue). The Filter Was Able To Produce Reliable Estimates of Diver's Position for Generating Trajectory Setpoints for the AUV.

3.5.3 *Elevation Mapping*

The process of building bathymetric (elevation) map is carried out in parallel and is fed to the trajectory tracker in order to estimate a target position for the AUV that not only is following the diver but is also at a desired height from the terrain. I used the method presented by Fankhauser *et al.* (2018) to generate elevation map using the point cloud obtained from the downward pointing stereo camera. An instance of the elevation map obtained with the sinusoidal terrain is shown in Fig. 3.5.

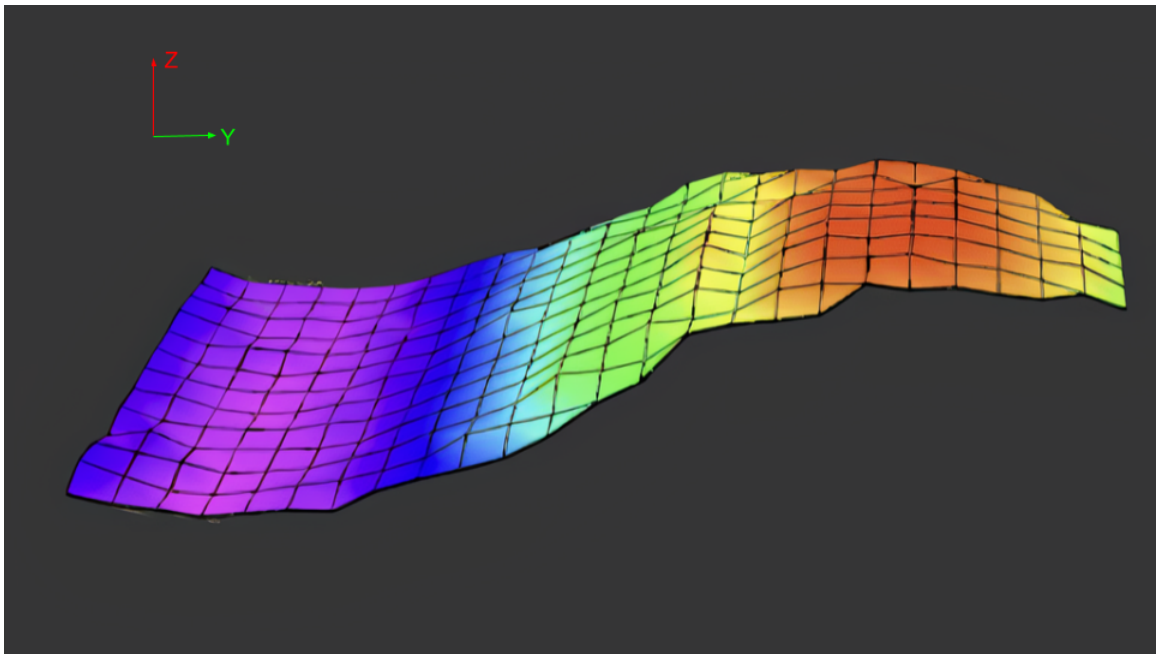

Figure 3.5: Robot-Centric Elevation Map That Was Generated in the Environment With the Sinusoidal Terrain. The Robot Uses This Map To Estimate the Depth of the Terrain and Accordingly Position the Setpoints of the Trajectory.

3.5.4 *Trajectory Generation*

The target position setpoint for the AUV in the world frame, $T^{W}_{R_{target}} = [x, y, z]$, is calculated at each time step using information from the diver tracking and elevation

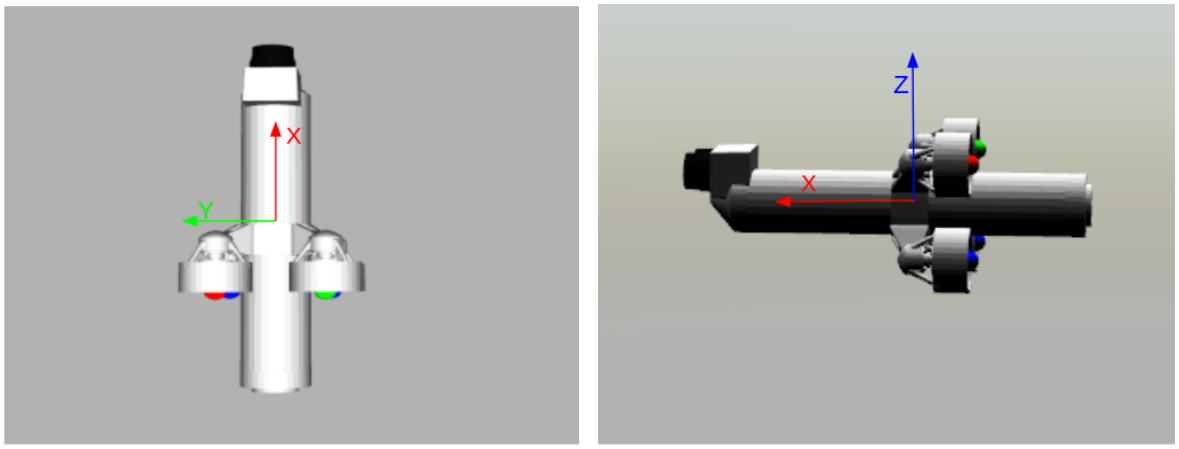

Figure 3.6: *HippoCampus* AUV Model in Gazebo That Was Used To Test the Pipeline. The Left Figure Shows the Top View and the Right Figure Shows the Side View.

mapping modules. The position vector T_D^R representing the diver's position in the robot's reference frame is obtained from the diver tracking module and provides the general direction that the robot needs to follow. I calculate the unit vector in this direction and scale it with a factor depending on the target velocity of the robot. The vector is then transformed to the world coordinate frame. The x, y coordinates of the vector so obtained constitute the corresponding coordinates of the target position. The z-coordinate of the target position is calculated using the elevation value at the corresponding x, y coordinates of the elevation map. More specifically, I calculate the depth of the terrain at the target position and add the distance to maintain from the terrain to it in order to obtain the z coordinate of the target position.

3.5.5 Position Controller

I used the *HippoCampus* AUV from Hackbarth *et al.* (2015) for testing the algorithm in Gazebo (shown in Fig. 3.6). The attitude controller for the AUV was

presented by Duecker *et al.* (2018) and was made available to be used with the PX4 flight stack. The position controller on top of this attitude controller takes in the current and desired position and velocity values of the robot and outputs the target thrust and orientation values to the attitude controller. The attitude controller from Hackbarth *et al.* (2015) takes care of transferring the motor commands to the simulated actuator in Gazebo.

In each iteration of the control loop of the position controller, the target force and orientation are calculated and sent to the attitude controller, which then generates motors speeds for the individual motors of the AUV. The approach to calculate thrust and desired orientation is based on the methods presented by Mohta *et al.* (2017). The force vector f required to accelerate the AUV towards target position p_{target} is given by:

$$\begin{aligned} e_{pos} &= p_{target} - p_{current} \\ e_{vel} &= v_{target} - v_{current} \\ f &= K_p e_{pos} + K_d e_{vel} \end{aligned} \tag{3.4}$$

where e_{pos} is the error between the target position, p_{target} and current position, $p_{current}$. e_{vel} denotes the error between the target velocity, v_{target} and the current velocity, $v_{current}$. K_p and K_d are the proportional and derivative gains respectively. The buoyancy is assumed to be equal to the weight of the robot.

In case of the *HippoCampus* AUV, the thrusters are co-planar and only provide force along the x-direction of the body frame. Hence in order to calculate the target thrust, the force obtained from eq. 3.4 is transformed into the body-fixed frame and only the x-component is extracted as shown in equation 3.5

$$thrust_{target} = R_w^b f \hat{e}_1 \tag{3.5}$$

where $R_b^w \in SO(3)$ denotes the transformation from the world frame to the body

frame of the robot and $\hat{e} = [1\ 0\ 0]$. The PX4 flight stack only accepts normalized thrust values between 0 and 1 and hence the thrust obtained is normalized to a value between 0 and 1, where 1 represents maximum thrust possible.

The thrust vector of the AUV is aligned with the x-axis of the body frame, hence the target orientation should be aligned with the force vector obtained from equation 3.4. Owing to the bidirectional nature of the propulsion system, there are two possible alignments. The target orientation $R_{des} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$, where $b_1, b_2, b_3 \in \mathbb{R}^3$ is calculated as

$$b_1 = sgn(thrust_{target}) \frac{f}{||f||} \tag{3.6}$$

where $sgn()$ denotes the signum function that is 1 or -1 depending on whether $thrust_{target}$ is positive or negative respectively.

The orientation of y-axis, b_2 can be calculated separately using the desired roll angle, θ for the robot as shown in equation 3.7 . I maintained $\theta = 0$ as I want the camera to be pointing downwards at all times. Once b_1 and b_2 are obtained, b_3 can be calculated as shown in equation 3.8. Since b_1 and b_2 are not necessarily orthogonal to each other, I recalculate b_2 as the cross multiplication of b_3 and b_1, as shown in equation 3.9

.

$$b_2 = \begin{bmatrix} 0 & cos\theta & sin\theta \end{bmatrix} \tag{3.7}$$

$$b_3 = \frac{b_1 \times b_2}{||b_1 \times b_2||} \tag{3.8}$$

$$b_2 = b_3 \times b_1 \tag{3.9}$$

3.6 Experiments

I tested the system in the Gazebo simulation environment and used the *UUV Simulator* plugin from Manhães *et al.* (2016) to simulate the hydrodynamic and hydrostatic effects. I evaluated the algorithm on three different synthetic terrains (i) Flat, (ii) Sinusoidal, and (iii) Coral Reef model. The Sinusoidal and Coral reef terrain models used in the experiments are shown in Fig. 3.7.

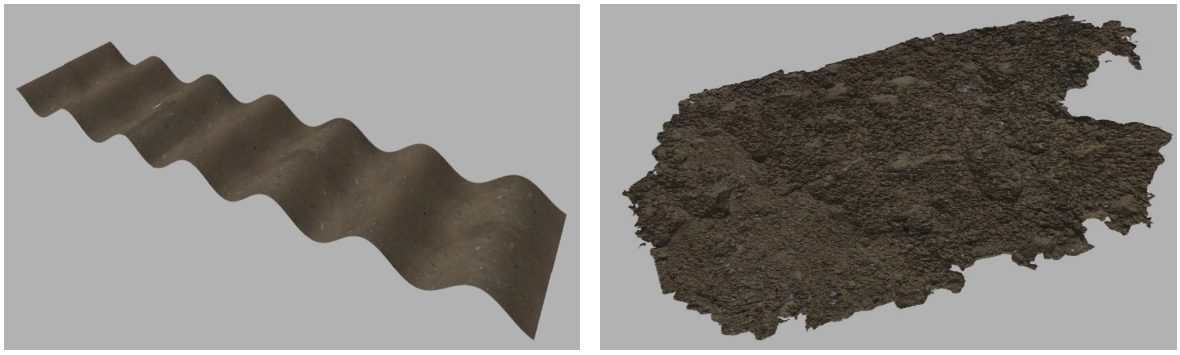

Figure 3.7: Terrain Models Used in the Experiments. Figure on the Left Shows the Sinusoidal Terrain That Was Generated in Blender Software from Community (2018). Figure on the Right Shows the Terrain Model of a Coral Reef in Honaunau Bay in Hawaii.

Although the flat terrain is overly simplified in comparison to the other two terrains, it validates the algorithm's ability to follow a diver. The diver was moved in sinusoidal motion according to the equation $y = 5sin(0.2x)$. Fig. 3.8 presents the y coordinates of the diver and the robot during the mission. As is evident from the figure, the robot moves accordingly in the sinusoidal fashion to follow the diver. Fig. 3.9 shows the z coordinate of the robot in the same experiment. The robot was initially at a lower depth ($-2m$) and attains a depth of around $-3.25m$ to roughly maintain a fixed distance of $1.75m$ from the terrain.

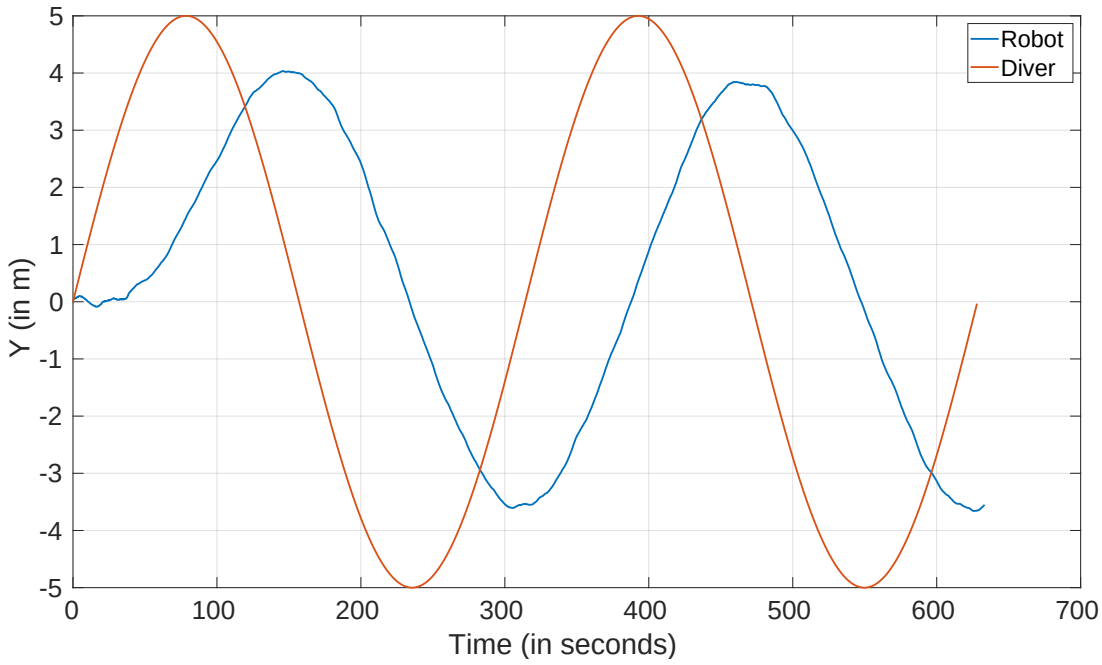

Figure 3.8: Diver Following on a Flat Terrain

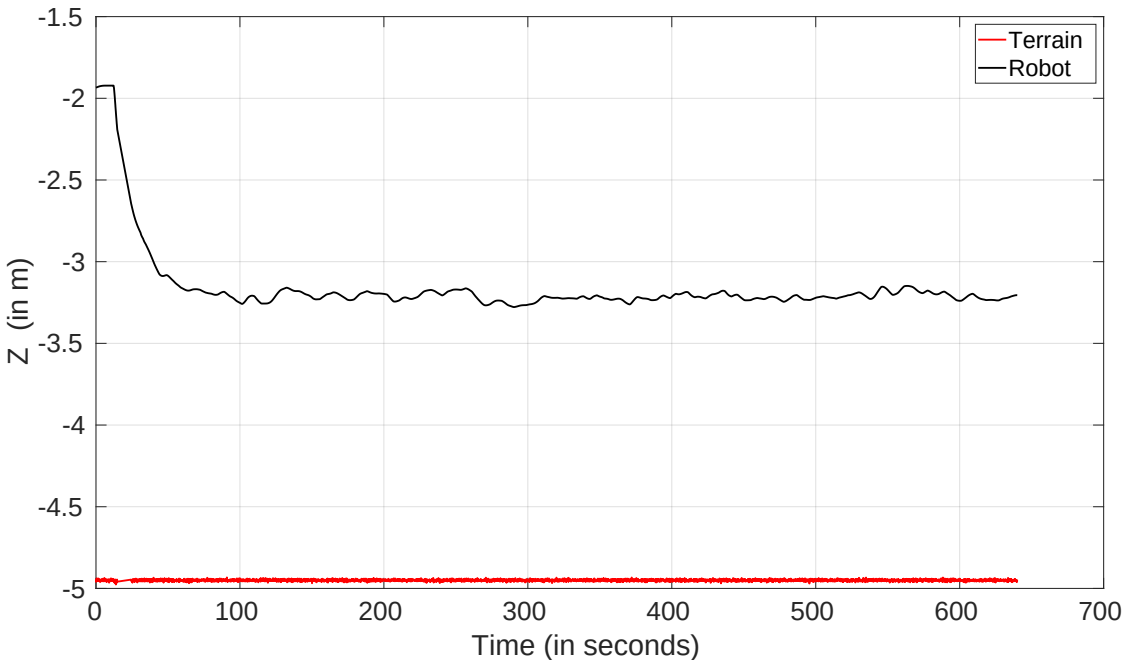

Figure 3.9: Terrain Following on a Flat Terrain

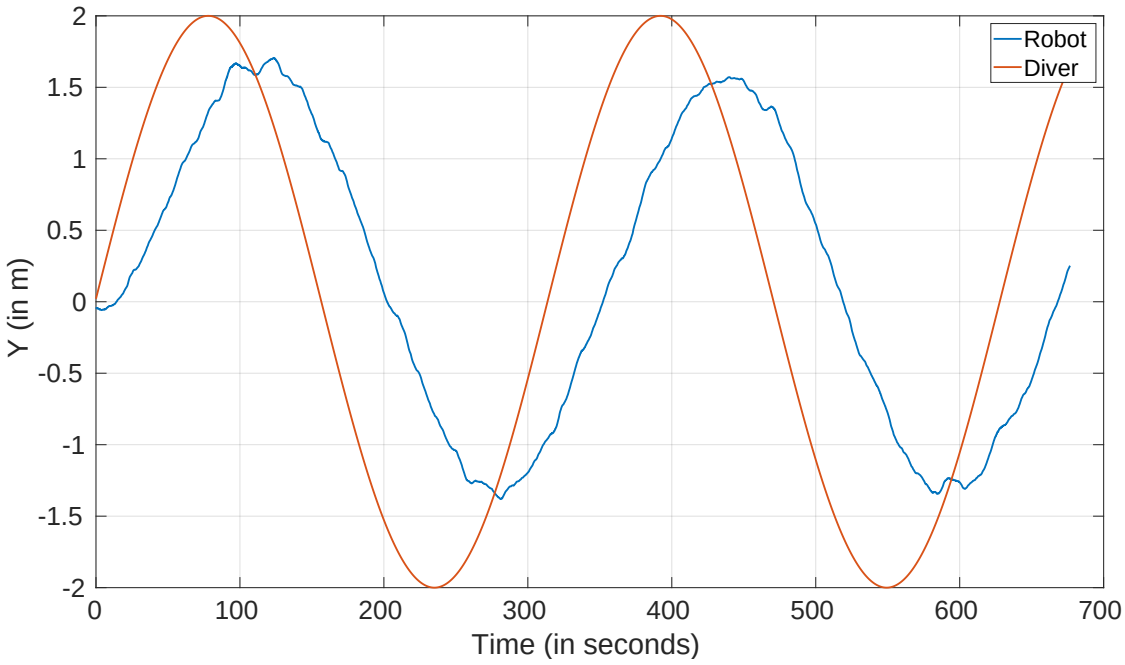

Figure 3.10: Diver Following on the Sinusoidal Terrain

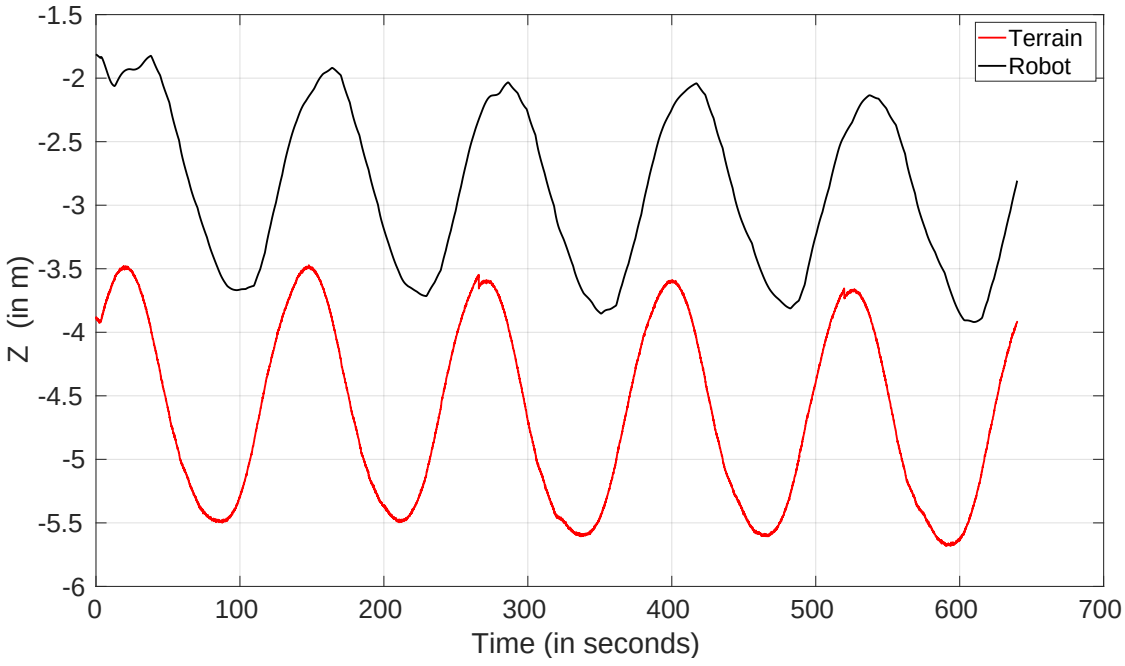

Figure 3.11: Terrain Following on the Sinusoidal Terrain

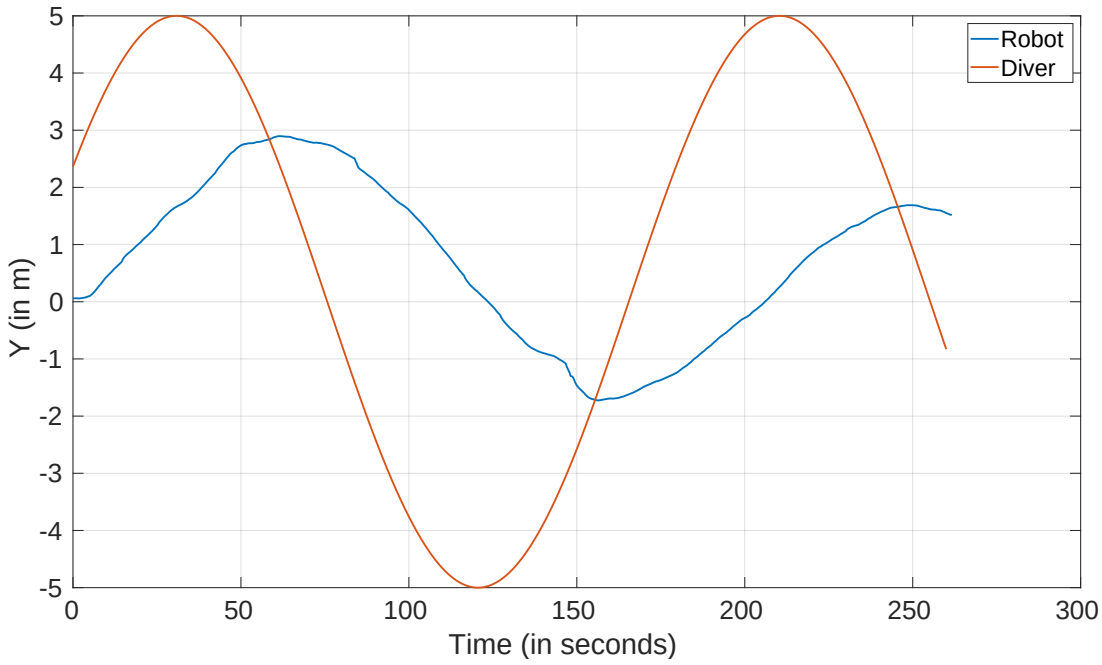

Figure 3.12: Diver Following on the Coral Reef Terrain

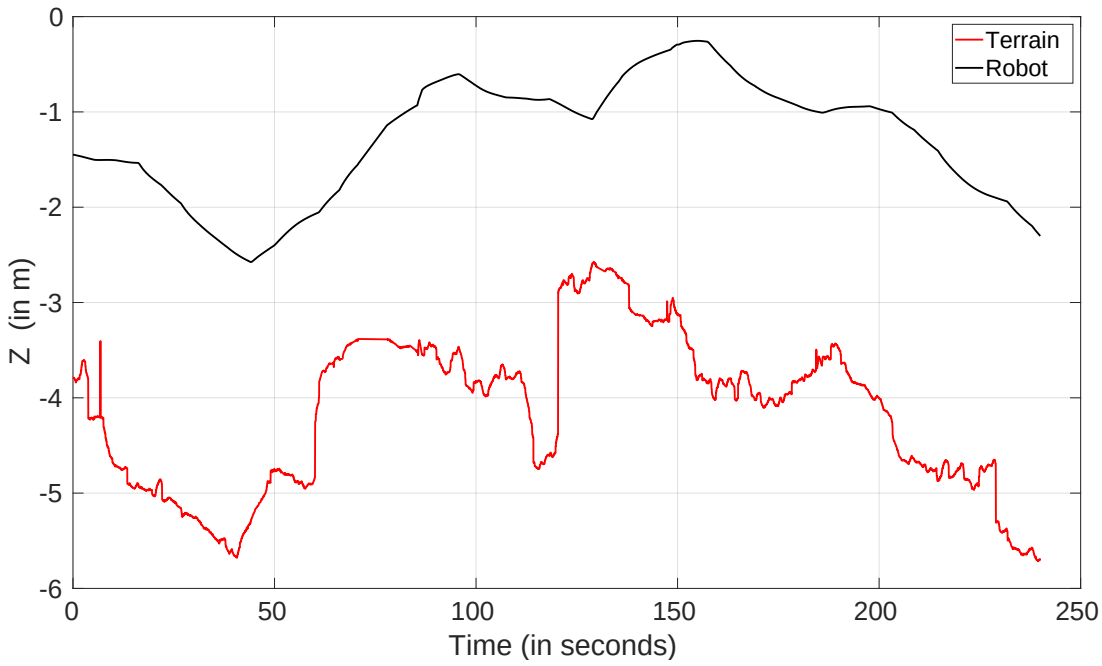

Figure 3.13: Terrain Following on the Coral Reef Terrain

The sinusoidal terrain presented a more challenging setting for both aspects of the system: *Terrain Following* and *Diver Following*. The challenge to the former aspect is obvious but to the latter is more subtle. Since the AUV has motors in a planar configuration, changing the elevation requires changing the pitch angle of the robot. This almost always leads to the diver moving out of the camera frame. This results in sparse measurement updates of the diver's position and makes the system rely on the estimates of diver's position from the *predict* step of the filter. I used a sinusoidal trajectory for the horizontal motion of the diver according to the equation $y = 2sin(0.2x)$ to test the tracking ability of the system. Fig. 3.10 shows the trajectories followed by robot and the diver. The robot was able to follow the diver in spite of the sporadic measurements. The terrain following on the sinusoidal terrain is shown using Fig. 3.11 which shows that the robot is able to maintain a fixed distance from the terrain.

Finally, I tested the system in an environment with the terrain model of a coral reef from Honaunau bay in Hawaii. This terrain presented a more realistic and challenging setting to test the algorithm as it had aribrary and sudden variations in depth. Fig. 3.12 shows the performance of diver following aspect of the algorithm in this terrain. The robot was able to keep track of the diver's position in spite of the sporadic measurement updates and was able to follow the diver throughout the mission.

The terrain following performance for this terrain is shown in Fig. 3.13. The roughness of the terrain did not severely affect the trajectory, thanks to the filtering of the point cloud done by the elevation mapping module. Fig. 3.13 shows that the robot roughly maintained a constant distance from the terrain.

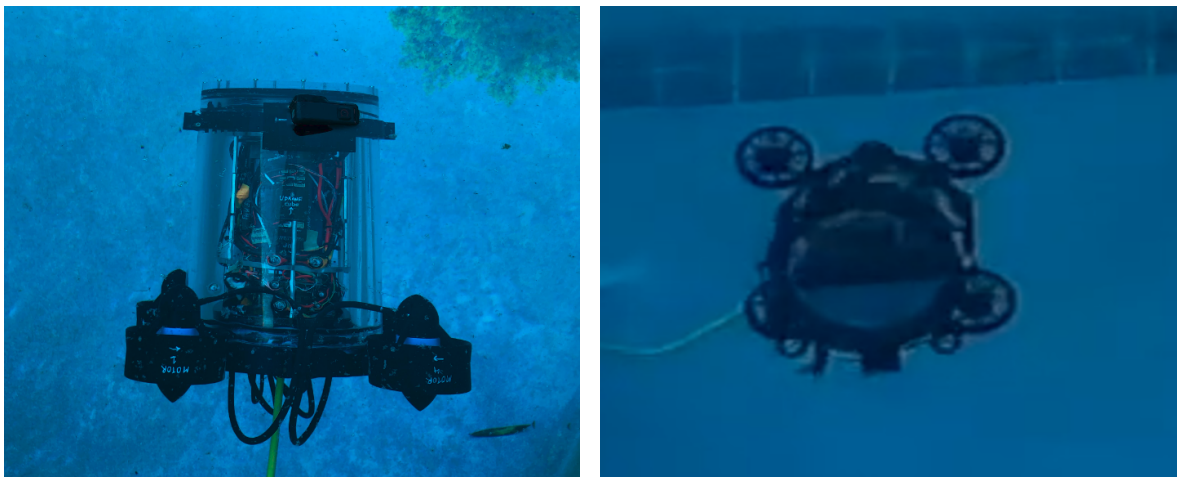

Figure 3.14: The Custom AUV, *uDrone* With a Planar Motor Configuration Similar to the *HippoCampus* AUV and Is Equipped With a Pixhawk 2.1, ZED2 Stereo Camera and NVIDIA Jetson TX2 Companion Computer. A Tether Is Used for Monitoring and Debugging, and Is Not Needed for Autonomous Operation.

3.7 Conclusion and Future Work

I presented a *Terrain-Relative Diver Following* algorithm that can enable an AUV to follow a human diver while maintaining a fixed distance from the terrain. I used a 3D model of a scuba diver to simulate a more realistic setting to test the algorithm. I presented the challenges associated with developing this system and presented the methods for addressing the same. The code and the model for diver detection are made available at github.com/algprasad/diver_follower. The future work includes conducting experiments on the custom AUV (shown in Fig. 3.14) which would require modifying the detection module of the pipeline.

Chapter 4

REAL TIME DENSE 3D SEMANTIC MAPPING

4.1 Introduction

Mapping is an essential quality of autonomous robots. In order to effectively operate in the real world, it is imperative that the robot has an accurate understanding of the environment. The past two decades have seen a proliferation of efficient mapping algorithms that can assist robots in localization (Mur-Artal *et al.* (2015), Engel *et al.* (2014), Schöps *et al.* (2019)), navigation (Hornung *et al.* (2013)), and reconstructing the environment (Wang *et al.* (2019), Labbé (2018)). However, a majority of the algorithms focus on providing a geometric understanding of the environment with the sole aim to enable ego-motion estimation. The ability to autonomously generate a map that not only encodes geometric information but also provides semantics about the environment can accelerate progress in a number of areas such as environmental monitoring, infrastructure inspection, and agriculture. Such a representation not only provides the domain experts with meaningful maps but also forms the basis of intelligent navigation algorithms to explore unknown environments. For instance, an underwater robot that can generate 3D semantic maps of the coral reef can provide quantitative estimates of the coral cover that could assist researchers in analyzing damage. At the same time, the semantic map generated in real time can help identify the interesting regions in the reef that need to be explored further. One other advantage of developing semantic maps as shown by McCormac *et al.* (2017) is that fusing multiple 2D predictions into a 3D map can directly lead to improvement in the 2D mask estimation as compared to baseline 2D segmentation predictions.

In this thesis, I present one such method to develop semantic 3D maps by fusing robot path, raw depth and RGB images. This was achieved by incorporating semantic information obtained by running segmentation algorithms on the RGB images into the dense surfel mapping method presented by Wang *et al.* (2019). The depth images are aligned to the RGB images to facilitate the correspondence between the pixel level classification and the 3D point cloud. That is, for each measurement of the depth image and the RGB image, a semantic point cloud is generated that has information about the class to which each point belongs. This is then fused with a local map extracted from a global representation using the pose information of the robot. The details of the semantic dense surfel mapping is presented in section 4.3.

The algorithm for incorporating semantic information presented in this thesis is robot agnostic. That is, the same algorithm can work for aerial, ground, and underwater robots. All that is needed is RGB image, corresponding depth image and the robot pose.

The rest of the thesis is organized as follows: Section 4.2 presents the related work in this domain. This is followed by section 4.3 that provides details on the Dense Semantic Surfel Mapping technique. The mathematical formulation for incorporation of semantics on top of the pipeline is given in section 4.4 followed by results in section 4.5.

4.2 Related Work

4.2.1 SemanticFusion

A seminal work in the field of real time dense 3D semantic mapping was presented by McCormac *et al.* (2017). Their system comprises of three main components: (i) Elasticfusion (Whelan *et al.* (2016)), a Simultaneous Localization And

Mapping (SLAM) system for obtaining pose information that is used to provide correspondences between multiple frames and also provides the backbone for a globally consistent map, (ii) a Convolutional Neural Network to predict per-pixel probabilities of the objects, and (iii) a Bayesian update scheme to update the probabilities based on new measurements.

Similar to the work presented in this thesis, a surfel-based representation was used for the map. A surfel (short for Surface Element) is a surface point that typically encodes the position, normal and radius of the unit elements in the model. For each new pose, the surfels observed are added to the map and the existing surfels are refined. The underlying SLAM system uses a combination of ICP presented by Besl and McKay (1992) and RGB alignment to estimate the robot pose.

Each surfel in the map encodes its position, normal and discrete probability distribution over set of classes. This probability is updated when new measurements of the surfel are observed using the equation 4.1

$$P\left(l_i \mid \boldsymbol{I}_{1,\ldots,k}\right) = \frac{1}{Z} P\left(l_i \mid \boldsymbol{I}_{1,\ldots,k-1}\right) P\left(O_{\boldsymbol{u}(s,k)} = l_i \mid \boldsymbol{I}_k\right) \tag{4.1}$$

where $l_i \in \mathbb{L}$ is the set of class labels, $\mathbb{O}_{\approxeq}$ is the observation for pixel u and I_k is the data for image k.

A major limitation of this approach is that of scalability. Although it works well for room scale environments, the system would not work well for large scale environmental mapping applications.

4.2.2 Meaningful Maps with Object-Oriented Semantic Mapping

Sünderhauf *et al.* (2017) presented a object-oriented approach to semantic mapping where they used a combination of SLAM, object detection, semantic segmentation to build a mapping system that maintains a list of objects detected in the map

and are stored as a collection of points clouds. The underlying SLAM algorithm is ORB-SLAM2 (Mur-Artal *et al.* (2015)), a popular SLAM algorithm that fuses RGB-D images to obtain pose estimates of the robot.

The object detection is done by Single-Shot Multi-box Detector (SSD) presented by Liu *et al.* (2016) that provides the bounding boxes for every keyframe. The 3D segmentation was done using a combination of both color and depth images to increase accuracy of segmentation.

Updating the map as new measurements arrive requires accurate data association and efficient update schemes. The former was addressed using a nearest neighbour search based on Euclidean distance. *k-d trees* was used for faster nearest neighbour queries. Each individual object contains the 3D point cloud, a vector with indices of all poses (from the pose graph) which observed the object, and a vector of confidences where the size of the vector is equal to the number of classes. The identity of the object is determined to be the class with largest normalized confidence.

4.2.3 Semantic 3D Mapping from Deep Image Segmentation

More recently, Martín *et al.* (2021) presented a 3D semantic mapping technique that uses image segmentation results from an RGB image and uses a voxel based model for representing the environment. The class of each voxel in the map are updated as new measurements arrive. This results in a semantic occupancy grid map which can be incredibly useful for intelligent motion planning algorithms. They also addressed the boundary pixel problem that arises when a segmented image is directly projected into the 3D space and the alignment between the RGB and Depth Images is inaccurate.

Although using a voxel based representation has its merits, the low 3D reconstruction resolution acts as a barrier for this approach to be used for environmental

monitoring.

4.3 Semantic Dense Surfel Mapping

I leveraged the dense surfel mapping technique presented by Wang *et al.* (2019) for my work and incorporated semantic information to surfels so as to obtain a semantic dense surfel map of the environment. Their algorithm is scalable and uses only CPU computation. Moreover, the addition of semantic information (a contribution of this thesis) to their pipeline adds very little overhead. For the sake of context and completeness, the following paragraphs describe the work presented in Wang *et al.* (2019). Section 4.4 will present the mathematical formulation for incorporating semantic information to the surfels which is the one of the contributions of this thesis.

The dense surfel mapping system presented by Wang *et al.* (2019) fuses information from depth images (obtained from RGB-D cameras, stereo cameras, monocular cameras or Solid State Lidars), intensity images and pose information (obtained from external sources or from the Visual odometry using RGB-D or RGB frames) to generate consistent 3D map of the environment using surfels. The choice of using surfels to represent the environment makes it convenient to fuse semantic information using semantic masks from RGB images.

An interesting feature of their system is that instead of representing every point from the point cloud as a surfel, they extracted super pixels from the depth and intensity image and then used a surfel to represent each super pixel. The super pixels were extracted using the technique presented in Achanta *et al.* (2012). A detailed description of this technique is presented in section 4.3. Assigning surfels to super pixels instead of pixels makes the algorithm scalable with lower loss in information as compared to sampling based reduction techniques. The use of super pixels is also conducive to using semantic masks as similar pixels are grouped together. Owing

to the similarity of these pixels, they are highly likely to belong to the same class. Section 4.3 contains information about their methods for local map extraction and fusion of surfels.

Super pixel extraction

The method used for extracting super pixels in their work is adapted from Achanta *et al.* (2010). The idea is to group pixels that are similar in intensity and depth values into one group in order to reduce the amount of redundant data while reducing the loss of information.

SLIC (Simple Linear Iterative Clustering) Algorithm generates super pixels in an image by grouping pixels that are similar in intensity and are spatially close. The similarity is calculated by calculating the Euclidean distance between the two pixels represented by a 5D vector. This 5D vector encodes the intensity information in the L^*a^*b color space and the spatial pixel coordinates.

The normalized distance between two pixels is calculated using the equations given in 4.2.

$$D_s = d_{lab} + \frac{m}{S} d_{xy} \tag{4.2}$$

where D_s is the distance measure, d_{lab} is the distance between 3D vectors representing the intensity of the pixels in l^*a^*b color space and is calculated as $d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$, m is parameter that is calculated empirically and is used to enforce the fact that spatial similarities are more important that intensity similarities, and S is the normalized grid interval and is equal to $\sqrt{N/K}$ where N is number of pixels in the original image and K denotes the number of super pixels.

In order to assign the pixels to a super pixel cluster, the super pixels seeds are initialized at regular intervals depending on the pre-defined number of super pixels.

Figure 4.1: Superpixels Generated in the Custom Image Using the SLIC Algorithm.

Then the pixels are moved to a position such that the gradient of intensity is zero. This is done to ensure that the super pixel centres are not assigned at an edge in the image. After that, the iterations of the commonly known *k-means* clustering are started wherein each iteration constitutes assigning all pixels to the nearest cluster centre such that the distance calculated using equation 4.2 is minimum. This is repeated until the convergence criteria is reached. An example of super pixels generated on a custom image is shown in figure 4.1 A comparison of SLIC with other state-of-the-art Super Pixel generation algorithms is presented in Achanta *et al.* (2012).

4.3.1 Surfel Fusion

For each super pixel in the RGB image, a surfel is initialized. The surfels used in their work are composed of $S = [S_p, S_n, S_c, S_w, S_r, S_t, S_i]$, where $S_p \in \mathbb{R}^3$ is the surfel position, $S_n \in \mathbb{R}^3$ is the surfel normal, $S_c \in \mathbb{R}$ is the mean cluster intensity, S_w is the

weight of the super pixel used in fusion, $S_r \in \mathbb{R}$ is the radius of the surfel, $S_i \in \mathbb{W}$ is the index of the reference key frame, $S_t \in \mathbb{W}$ is the number of times the surfel has been fused by other frames.

In order to fuse the newly initialized surfels with the surfels in the map, the relevant surfels need to be extracted from the map. This is done by using the pose graph and only extracting the surfels that are attached to the pose index. This results in O(1) update time regardless of the size of the global map leading to high scalability. The locally consistent keyframes are found using breadth first search technique on the pose graph.

Once the local map is extracted, the data association is done by back-projecting the local surfels into the input frame and if they have similar depth and normal, local surfel is fused with the new surfel using the equations given in 4.3

$$\begin{aligned} S^l_{\mathrm{p}} &\leftarrow \tfrac{S^l_{\mathrm{p}} S^l_w + S^n_{\mathrm{p}} S^n_w}{S^l_w + S^n_w}, S^l_c \leftarrow S^n_c \\ S^l_{\mathbf{n}} &\leftarrow \tfrac{S^l_{\mathrm{n}} S^l_w + S^n_{\mathrm{n}} S^n_w}{S^l_w + S^n_w}, S^l_i \leftarrow S^n_i \\ S^l_t &\leftarrow S^l_t + 1, \quad S^l_w \leftarrow S^l_w + S^n_w \\ S^l_r &\leftarrow \min\left(S^n_r, S^l_r\right). \end{aligned} \tag{4.3}$$

where S^l_{p} and S^n_{p} are the positions of the local and the new surfel, S^l_w and S^n_w are the weights of the local and the new surfels, S^l_c and S^n_c denote the intesities of the local and the new surfels, S^l_r and S^n_r denote the radii of the local and new surfels.

4.4 Incorporating Semantic Information

The semantic information is provided in the form of semantic mask images where the pixels that belong to the class of interest are represented with black color. This provides the measured value for each surfel which is refined using a probabilistic framework. This framework facilitates fusing noisy semantic values into the dense surfel map and obtain real time object segmentation using segmentation from the

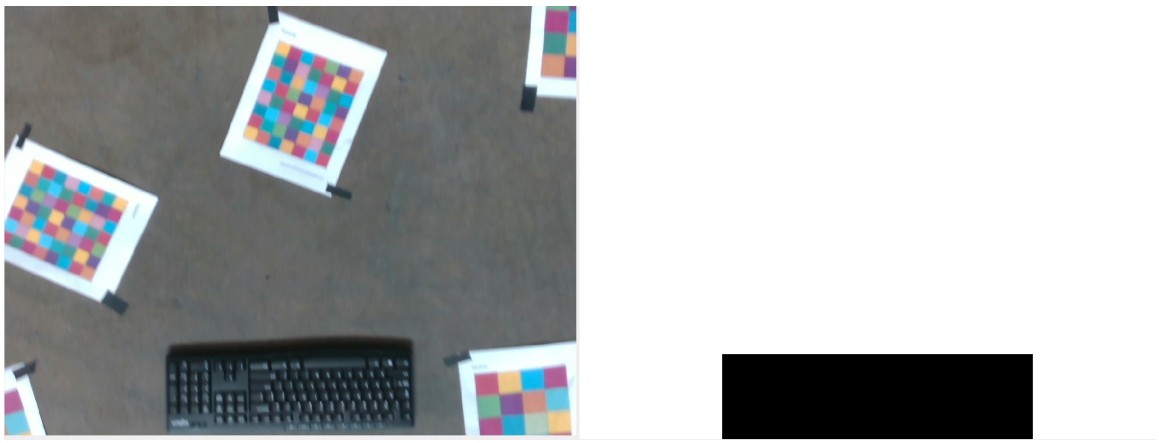

Figure 4.2: Raw Image (Right) and the Semantic Binary Mask for Keyboard (Left). This Binary Mask Is Then Assigned to the Surfel Which Is Updated Probabilistically As New Measurements Arrive

RGB images. Each surfel is now represented as $S = [S_p, S_n, S_c, S_r, S_t, S_i, S_m]$, where S_m is the mask of the surfel initialized by the semantic segmentation pipeline and defines the class to which the surfel belongs. A typical example of mask generated from the RGB image is shown in figure 4.2.

The mathematical formulation for incorporating the semantic information into the dense surfel map is inspired from the mathematical formulation of the occupancy grid map presented in Thrun *et al.* (2005), Moravec and Elfes (1985), and Stachniss (2021).

4.4.1 Problem Formulation

Given the measured mask values of each surfel z_i and pose of the camera x_i, estimate the surfel map m^* such that

$$m* = argmax_m P(m|x_1, z_1, ...x_t, z_t); \tag{4.4}$$

Assumptions

- Each surfel belongs to only one particular class.
- The surfel map remains static
- There is no dependency between the neighbouring surfels.

4.4.2 Formulation

The map m can be represented as the joint probability distribution $p(m)$ as:

$$p(m) = p(m_1, m_2, ...m_N); \tag{4.5}$$

where $m_1, m_2...$ represent the probabilities of each individual surfel. This can be represented as

$$p(m) = \prod_i p(m_i) \tag{4.6}$$

Given sensor data $z_{1:t}$ and the poses $x_{1:t}$ of the sensor, the map can be estimated as

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{4.7}$$

Since it is a binary random variable, a binary bayes filter for a static state can be employed. Following the formulation of occupancy grid map presented in Thrun *et al.* (2005), Moravec and Elfes (1985), and Stachniss (2021),

$$p(m_i|z_{1:t}, x_{1:t}) = \left[1 + \frac{1 - p(m_i|z_t, x_t)}{p(m_i|z_t, x_t)} \frac{1 - p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i|z_{1:t-1}, x_{1:t-1})} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1} \tag{4.8}$$

To make the formulation more efficient, *log odds* notation is used, which is obtained by taking the logarithm on both sides of equation 4.8. This would result in equation 4.9

$$l(m_i|z_{1:t}, x_{1:t}) = l(m_i|z_t, x_t) + l(m_i|z_{1:t-1}, x_{1:t-1}) - l(-m_i) \tag{4.9}$$

Analogously,

$$l_{t,i} = inverse\_measurement\_model(m_i, x_t, z_t) + l_{t-1,i} + l_0 \tag{4.10}$$

Inverse Measurement Model for Semantic Masks

In order to incorporate classification measurements obtained from the masks of the RGB images, the correspondence between the measured mask and the probability has to be established. This is usually based on the confidence with which the mask is predicts the image segmentation, which has to be calibrated based on the segmentation training results. For this work, I assumed that if a particular pixel is predicted to belong to a particular category, the probability of the prediction is 0.95 and hence the log odds is 2.994. Similarly, for every pixel that does not belong the class, the probability of prediction is 0.05 and the log odds is -2.994.

4.5 Results

The code for fusing semantic information to a dense surfel map was written on top of dense surfel mapping (Wang *et al.* (2019)). I tested the algorithm in a simulated environment in Gazebo as shown in the figure 4.3.

The raw image, super pixels generated and the corresponding semantic mask for this environment are shown in figure 4.4.

For the simulated setting, I used a colour based segmentation mask wherein the RGB image would generate binary masks for objects that are blue in colour. This segmentation mask image is then fused with the rest of the dense surfel mapping pipeline to generate surfels that encode the information of the object class to which they belong. This mask information is then fused with new information from the new measurements using equation 4.10. Hence the surfels get updated in terms of their position, normal and also class. This leads to simple and efficient way of visualizing

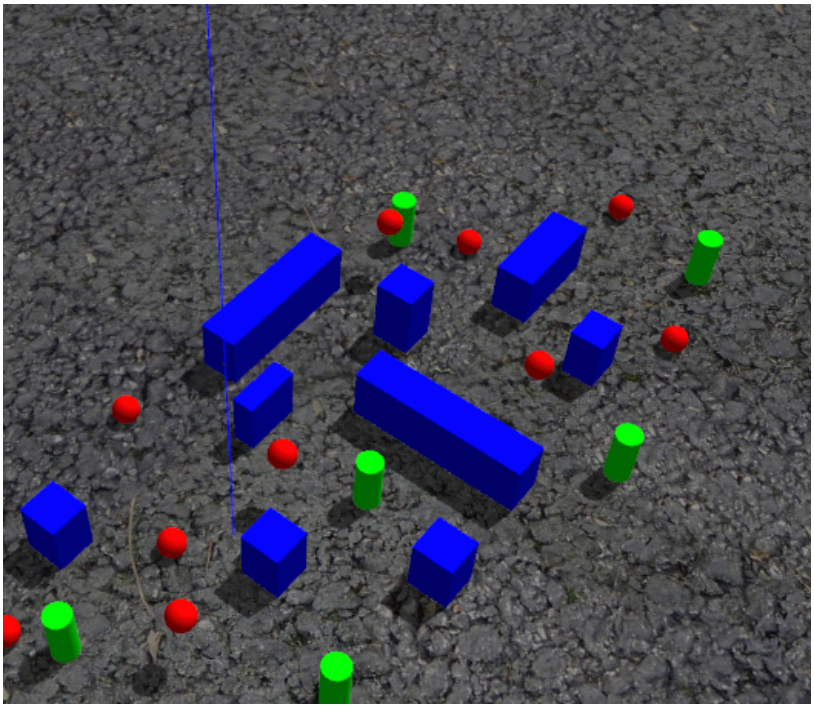

Figure 4.3: Simulated World in Gazebo With Different Object for Testing Dense Semantic Surfel Mapping Algorithm

the surfels that belong to the objects of interest. The figure 4.5 shows the results of the semantic segmentation where the objects were segmented and visualized as a different point cloud.

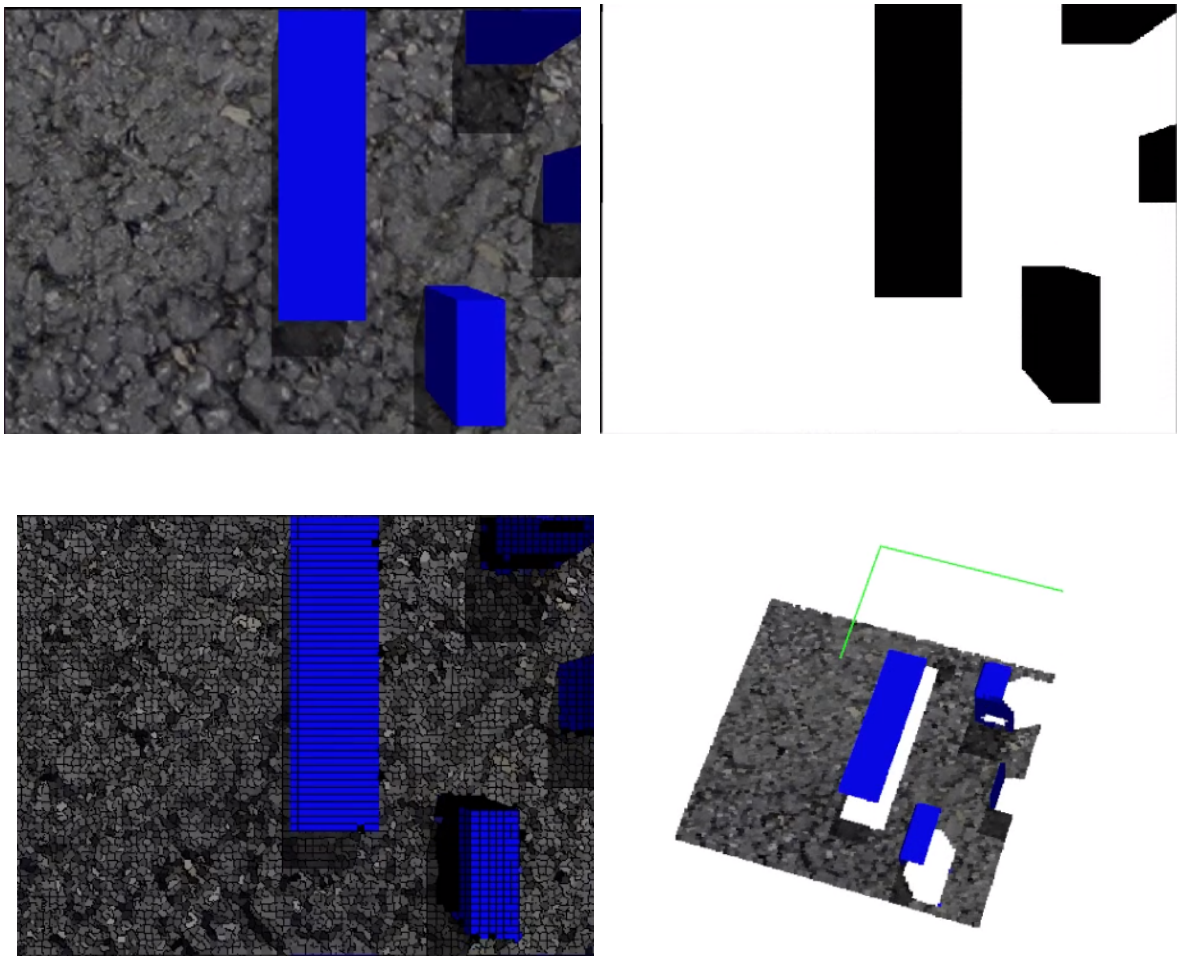

Figure 4.4: Top Left Shows the Raw Color Image From the Simulated Camera. Top Right Shows That Segmented Image Generated by Processing Only the Raw Color Image. Bottom Left Shows the Super Pixels Generated in the Image. Bottom Right Shows the Dense Surfel Map Generated Using the Algorithm.

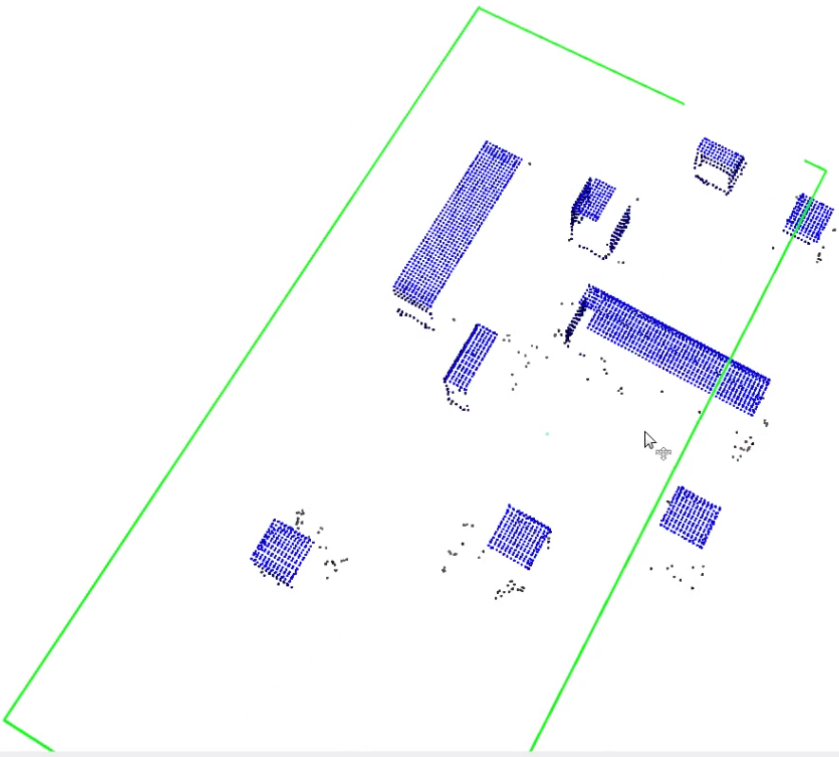

Figure 4.5: Results of a Semantic Dense Surfel Map in the Gazebo Simulation Environment. The Blue Points Are All the Segmented Surfels That Were Assigned the Class Based on the Semantic Mask. The Green Lines Show the Path Taken by the Camera

Chapter 5

CONCLUSION

This thesis presented efficient navigation and semantic mapping algorithms that can enable autonomous robots to navigate in complex environments and generate meaningful maps of the environment. The first part of my thesis presented a terrain relative navigation approach that can maintain a fixed distance from the terrain while following a human diver. This would facilitate collecting consistent data while visiting important areas in the reef that the expert diver wants to map closely. The second part of my thesis presented a dense semantic surfel mapping technique that encodes object classification information from the RGB imagery into surfels that represent the environment. This would enable extracting meaning from the surfel cloud that can not only present semantic data in real time but also has the potential to assist in more informed path planning as part of sophisticated exploration algorithms. The navigation and mapping algorithms were tested in a simulated setting in Gazebo. The results were analyzed and the code for the navigation algorithm has been made available.

5.1 Future Work

There are multiple directions of research that can be pursued to take the research presented in this thesis to the next level. Firstly, the Terrain Relative Navigation algorithm can be tested in the in-house pool test-bed. This would require adding a pre-processing step for the camera data to account for the refraction of light passing through multiple media in the underwater setting.

Nonlinear Model Predictive Control can be tested on the *uDrone*. This would

enable tracking even smoother trajectories and consequently improve the data quality. As the *uDrone* is based on PX4 flight stack, transitioning to optimal controllers would only require changing the top level controller keeping the rest of modules intact.

Semantic Dense Surfel Mapping can be improved to incorporate multiple object classes instead of a binary mask. This would require modifying the formulation to include multiple masks from the semantic segmentation on the RGB images.

REFERENCES

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "Slic superpixels", Technical report, EPFL (2010).

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods", IEEE Transactions on Pattern Analysis and Machine Intelligence **34**, 11, 2274–2282 (2012).

Besl, P. J. and N. D. McKay, "Method for registration of 3-D shapes", in "Sensor Fusion IV: Control Paradigms and Data Structures", edited by P. S. Schenker, vol. 1611, pp. 586 – 606, International Society for Optics and Photonics (SPIE, 1992), URL `https://doi.org/10.1117/12.57955`.

Bjelonic, M., "YOLO ROS: Real-time object detection for ROS", `https://github.com/leggedrobotics/darknet_ros` (2016–2018).

Community, B. O., *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, URL `http://www.blender.org` (2018).

Duecker, D. A., A. Hackbarth, T. Johannink, E. Kreuzer and E. Solowjow, "Micro underwater vehicle hydrobatics: A submerged furuta pendulum", in "2018 IEEE International Conference on Robotics and Automation (ICRA)", pp. 7498–7503 (2018).

Edge, C., S. S. Enan, M. Fulton, J. Hong, J. Mo, K. Barthelemy, H. Bashaw, B. Kallevig, C. Knutson, K. Orpen and J. Sattar, "Design and experiments with loco auv: A low cost open-source autonomous underwater vehicle", (2020).

Engel, J., T. Schöps and D. Cremers, "Lsd-slam: Large-scale direct monocular slam", in "Computer Vision – ECCV 2014", edited by D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, pp. 834–849 (Springer International Publishing, Cham, 2014).

Fankhauser, P., M. Bloesch, C. Gehring, M. Hutter and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates", in "International Conference on Climbing and Walking Robots (CLAWAR)", (2014).

Fankhauser, P., M. Bloesch and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization", IEEE Robotics and Automation Letters (RA-L) **3**, 4, 3019–3026 (2018).

Gockley, R., J. Forlizzi and R. Simmons, "Natural person-following behavior for social robots", in "Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction", HRI '07, p. 17–24 (Association for Computing Machinery, New York, NY, USA, 2007), URL `https://doi.org/10.1145/1228716.1228720`.

Hackbarth, A., E. Kreuzer and E. Solowjow, "Hippocampus: A micro underwater vehicle for swarm applications", in "2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 2258–2263 (2015).

Hornung, A., K. M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees", Autonomous Robots URL https://octomap.github.io, software available at https://octomap.github.io (2013).

Houska, B., H. Ferreau and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization", Optimal Control Applications and Methods **32**, 3, 298–312 (2011).

Islam, M. J., M. Fulton and J. Sattar, "Toward a generic diver-following algorithm: Balancing robustness and efficiency in deep visual detection", IEEE Robotics and Automation Letters **4**, 1, 113–120 (2019).

Islam, M. J., J. Hong and J. Sattar, "Person-following by autonomous robots: A categorical overview", The International Journal of Robotics Research **38**, 14, 1581–1618, URL https://doi.org/10.1177/0278364919881683 (2019).

Johnson, A. E. and J. F. Montgomery, "Overview of terrain relative navigation approaches for precise lunar landing", in "2008 IEEE Aerospace Conference", pp. 1–10 (2008).

Kalman, R. E., "A new approach to linear filtering and prediction problems", Transactions of the ASME–Journal of Basic Engineering **82**, Series D, 35–45 (1960).

Kamel, M., M. Burri and R. Y. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles", ArXiv **abs/1611.09240** (2016).

Koenig, N. and A. Howard, "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)", vol. 3, pp. 2149–2154 (IEEE, 2004).

Krukowski, S. and S. Rock, "Waypoint planning for autonomous underwater vehicles with terrain relative navigation", in "OCEANS 2016 MTS/IEEE", (Monterey, CA, 2016), URL https://web.stanford.edu/group/arl/sites/default/files/public/publications/KrukowskiR2016.pdf.

Labbé, M., "Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation", (2018).

Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu and A. C. Berg, "Ssd: Single shot multibox detector.", in "ECCV (1)", edited by B. Leibe, J. Matas, N. Sebe and M. Welling, vol. 9905 of *Lecture Notes in Computer Science*, pp. 21–37 (Springer, 2016), URL http://dblp.uni-trier.de/db/conf/eccv/eccv2016-1.html#LiuAESRFB16.

Manhães, M. M. M., S. A. Scherer, M. Voss, L. R. Douat and T. Rauschenbach, "Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation", in "OCEANS 2016 MTS/IEEE Monterey", pp. 1–8 (2016).

Martín, F., F. González, J. M. Guerrero, M. Fernández and J. Ginés, "Semantic 3d mapping from deep image segmentation", Applied Sciences **11**, 4, URL https://www.mdpi.com/2076-3417/11/4/1953 (2021).

McCormac, J., 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)", pp. 4628–4635 (2017).

Meduna, D., S. M. Rock and R. McEwen, "Closed-loop terrain relative navigation for auvs with non-inertial grade navigation sensors", in "IEEE-OES Autonomous Underwater Vehicles Conference (AUV)", (Monterey, CA, 2010), URL http://www.stanford.edu/group/arl/cgi-bin/drupal/sites/default/files/public/publications/MedunaRM_2010.pdf.

Meduna, D. K., S. M. Rock and R. McEwen, "Low-cost terrain relative navigation for long-range auvs", in "OCEANS 2008", pp. 1–7 (2008).

Meier, L., D. Honegger and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms", in "2015 IEEE International Conference on Robotics and Automation (ICRA)", pp. 6235–6240 (2015).

Melo, J. and A. Matos, "Survey on advances on terrain based navigation for autonomous underwater vehicles", Ocean Engineering **139**, 250–264, URL https://www.sciencedirect.com/science/article/pii/S002980181730241X (2017).

Mohta, K., M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Z. Zhu, J. A. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments", CoRR **abs/1712.02052**, URL http://arxiv.org/abs/1712.02052 (2017).

Moravec, H. and A. Elfes, "High resolution maps from wide angle sonar", in "Proceedings. 1985 IEEE International Conference on Robotics and Automation", vol. 2, pp. 116–121 (1985).

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

Müller, J., C. Stachniss, K. O. Arras and W. Burgard, "Socially inspired motion planning for mobile robots in populated environments", (2008).

Nsfr750 and CGTrader, "Diver: 3d model, [online]. available: https://www.cgtrader.com/free-3d-models/character/man/diver-7b35360b244e5441b29f4b0483bd1701", URL https://www.cgtrader.com/free-3d-models/character/man/diver-7b35360b244e5441b29f4b0483bd1701 (2014).

Petillot, Y., S. Reed and J. Bell, "Real time auv pipeline detection and tracking using side scan sonar and multi-beam echosounder", Oceans Conference Record **1**, 217–222, ocean's 2002 Conference and Exhibition ; Conference date: 29-10-2002 Through 31-10-2002 (2002).

Redmon, J., S. K. Divvala, R. B. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection", CoRR **abs/1506.02640**, URL `http://arxiv.org/abs/1506.02640` (2015).

Schöps, T., T. Sattler and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam", in "2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 134–144 (2019).

Stachniss, C., "Occupancy grid maps (cyrill stachniss)", (2021).

Stanford Artificial Intelligence Laboratory et al., "Robotic Operating System", URL `https://www.ros.org` (2018).

Sünderhauf, N., 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)", pp. 5079–5085 (2017).

Thrun, S., W. Burgard and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)* (The MIT Press, 2005).

Vaganay, J., M. Elkins, D. Esposito, W. O'Halloran, F. Hover and M. Kokko, "Ship hull inspection with the hauv: Us navy and nato demonstrations results", in "OCEANS 2006", pp. 1–6 (IEEE, 2006).

Wang, K., F. Gao and S. Shen, "Real-time scalable dense surfel mapping", arXiv preprint arXiv:1909.04250 (2019).

Whelan, T., R. F. Salas-Moreno, B. Glocker, A. J. Davison and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation", The International Journal of Robotics Research **35**, 1697 – 1716 (2016).