Semidefinite Programming Gradient Descent vs. SLAM: A Comparative Analysis for Reconstructing Seabed Terrain

P.K. Kartoidjojo

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Semidefinite Programming Gradient Descent vs. SLAM: A Comparative Analysis for Reconstructing Seabed Terrain

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

P.K. Kartoidjojo

May 23, 2025

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical Engineering (ME) for acceptance a thesis entitled

Semidefinite Programming Gradient Descent vs. SLAM: A Comparative Analysis for Reconstructing Seabed Terrain

by

P.K. Kartoidjojo

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: May 23, 2025

Supervisor(s):

prof.Dr. M. (Manuel) Mazo Espinosa

S.Vakili

Reader(s):

prof.Dr. M. (Manuel) Mazo Espinosa

D. (Dimitris) Boskos

S.Vakili

Table of Contents

Acknowledgements

I would like to express my deepest gratitude to my supervisor, prof.Dr. M. (Manuel) Mazo Espinosa, for his invaluable guidance, unwavering patience, and compassionate understanding of mental health throughout this project. Secondly, I would like to thank S. Vakili for his guidance at both the beginning and end of my thesis. Although we experienced some ups and downs in our communication, he played a key role in shaping the early stages of this work, for which I am sincerely grateful. I would also like to thank my girlfriend, Miriam Kebernik, for her constant encouragement and support throughout every stage of my thesis. Her belief in me has meant more than words can express.

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the Marie Skłodowska-Curie Grant under Agreement 956200.

Delft, University of Technology May 23, 2025

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) is a technology in robotics and autonomous systems that enables to build a map of an unknown environment while simultaneously tracking its own position within it. Originally developed for robotic navigation, SLAM has since become a solution applied across many domains, including robotics, medicine, agriculture, archaeology, and infrastructure management.

On of the application where SLAM is applied is on Autonomous Underwater Vehicles (AUVs). AUVs are essential for various underwater tasks, from scientific research to military activities. These tasks requires AUVs to navigate and map the underwater environment with a certain degree of precision. However, AUVs encounter significant difficulties due to the absence of GPS signals underwater, communication limitations, and changing environmental conditions.

Vakili, Sasan, et al. [1] proposed an algorithm aimed at estimating the unknown time-varying output map of a linear time-varying system using a Maximum A Posteriori (MAP) estimation approach. This algorithm employs Bayesian principles to integrate prior information and utilizes convex optimization and gradient-based techniques for fast and precise parameter estimation. One of the claims of the paper is that the Semi-definite Programming Gradient Descent(SDP-GD)can be deployed in AUV for seabed reconstruction.

Objective

In this work, the SDP-GD algorithm is adopted as a novel SLAM approach and evaluated against two established SLAM techniques: ORB-SLAM2—a state-of-the-art visual SLAM system—and a sonar-based SLAM method. The goal is to assess the viability and performance of SDP-GD in underwater mapping scenarios by comparing it to these conventional alternatives. For simplification, the trajectory is constrained to motion along the x-axis while maintaining a steady depth along the z-axis.

The evaluation is conducted within a custom-built simulation environment that includes realistic models of Autonomous Underwater Vehicles (AUVs) and their surrounding environments. Specifically, the *BlueROV2* is used to simulate visual SLAM with ORB-SLAM2, while the *DestiK SAGA* platform is simulated for sonar-based SLAM. To enhance realism, various forms of noise are introduced in the environment to mimic real-world underwater disturbances.

A visual SLAM algorithm is included in the comparison due to its relevance in aerial robotics—such as drones—which exhibit similar motion characteristics to underwater vehicles in certain scenarios. Conversely, the sonar-based SLAM method is chosen for its widespread application in underwater robotics, where acoustic sensing is often more robust in low-visibility conditions.

Scope

Through this structure, the report aims to assess the potential of the SDP-GD algorithm as a viable SLAM solution for underwater applications. Its performance is benchmarked against conventional SLAM approaches under realistic, simulated conditions.
The report[1] begins by presenting the theoretical background, including the core tasks of SLAM, an overview of existing SLAM techniques, and a detailed introduction to the SDP-GD framework.

The remainder of the report presents the original research conducted in this study. It begins with a description of the system setup, focusing on the Gazebo simulation environment and the configuration of both visual SLAM (using ORB-SLAM2) and sonar-based SLAM systems. Next, the design of the PID controller used for AUV control within the simulation is discussed. This includes analysis of its performance, response to disturbances, and integration with the SLAM systems to ensure stable navigation.
Since the SDP-GD algorithm requires a discrete-time, time-varying system model—while most AUV models are defined in continuous time—the report also addresses the modeling and system identification process. This includes nonlinear modeling, system linearization, and closed-loop system identification techniques to obtain a suitable model for use with SDP-GD. Following this, the adaptation of the SDP-GD algorithm for both sonar-based and visual SLAM inputs is explained in detail. A comprehensive comparative evaluation is then conducted, assessing the performance of SDP-GD against ORB-SLAM2 and the sonar-based SLAM method. Key evaluation metrics include depth estimation accuracy, computational efficiency, effects of sampling frequency, and robustness to environmental disturbances.

[1]Parts of this thesis were reviewed and refined using ChatGPT to improve grammar, clarity, and conciseness. Some sections were summarized to shorten the text, but no new content or ideas were generated by the tool. All technical work, analysis, and conclusions were developed entirely by the author.

Chapter 2

Theoretical Background

This chapter outlines the key concepts used in this thesis. It starts with the basics of Simultaneous Localization and Mapping (SLAM), explaining how it works and some common methods. Next, it covers the main sensors used for underwater navigation, including cameras, sonar, inertial measurement units (IMUs), and Doppler Velocity Logs (DVLs). Finally, it introduces the SDP-GD framework, an algorithm that estimates the unknown, time-varying output map of a linear system using a Maximum A Posteriori (MAP) approach.

2-1 Two Core Tasks of SLAM

Simultaneous Localization and Mapping (SLAM) is the process of building a map of an unknown environment while simultaneously estimating the location of a moving system within that environment [2]. SLAM is particularly useful when GPS is unavailable or unreliable, such as underwater or in space.

As the name suggests, SLAM combines two key tasks: mapping the environment and localizing the system within it. These tasks are closely connected — as the system collects sensor data and updates its map, it also improves its estimate of its own position. In certain algorithm this loop continues as the robot moves, gradually refining both the map and its position estimate. In this way, the robot is able to navigate through unknown environments autonomously.

Localization is the process of estimating the robot's state — such as position, orientation, and velocity — relative to its environment. Localization can be done using sensor data such as visual odometry, GPS, or inertial measurements. When sensors rely on environmental observations (e.g. visual odometry), the poses of known landmarks are used to estimate the robot's location.

Mapping refers to the process of creating a structured representation of the environment, often in the form of a 2D or 3D map [3]. This can be done by estimating the system's state and updating the map based on measurements collected by the sensors. In visual slam he environment is typically represented using *landmarks* or *features*, each with its own estimated pose (position and orientation). The poses of the landmarks can be used to update the map.

2-2 Overview of SLAM Techniques

In this thesis, one of the comparisons involves evaluating visual SLAM and sonar-based SLAM, particularly in the context of underwater navigation. To better understand the differences between the two, the following sections provides an overview of both methods.

Visual SLAM

Visual SLAM relies on visual input to estimate motion and map the environment. It includes pure visual approaches and **Visual-Inertial SLAM**, which incorporates IMU data. *Taketomi et al.* [4] identify three core modules: initialization, tracking, and mapping. These maintain consistent pose and map estimates. A key concept is odometry—estimating motion over time. Visual odometry (VO) uses sequential camera images, as opposed to traditional odometry sensors, to maintain geometric consistency. **Monocular visual** SLAM faces challenges due to limited depth and field of view [4], making initialization and scale estimation difficult. To mitigate this, this thesis employs a stereo camera setup, enabling depth estimation from image disparity. **Visual SLAM methods** are classified into *feature-based* and *direct*. This thesis adopts a feature-based approach, detailed below.

Feature-Based Methods: These methods track distinct features (e.g., corners, edges) across frames to estimate motion and reconstruct the 3D environment. PTAM, a notable example, introduced parallel tracking and mapping and uses a five-point algorithm for initialization [5], unlike MonoSLAM which requires prior geometry.

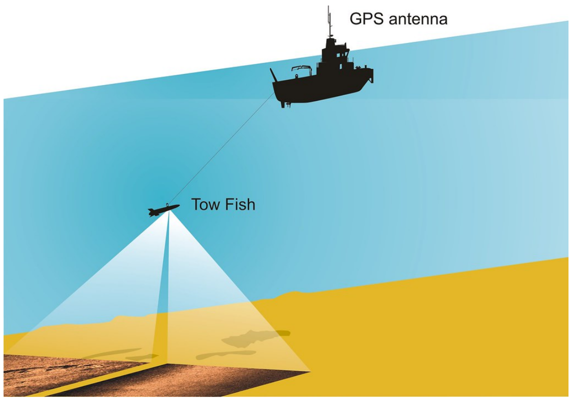
However, they depend on textured scenes. In low-texture environments—like underwater—feature extraction is unreliable, reducing localization accuracy [6].

Sonar-Based SLAM

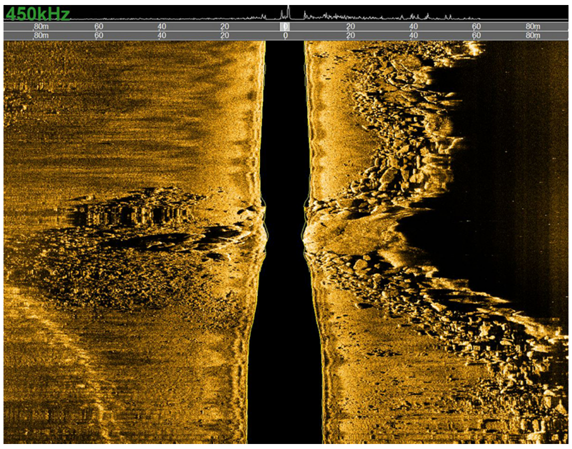
Sonar-Based SLAM is a technique for estimating a vehicle's trajectory and simultaneously building a map of the underwater environment using sonar data. Unlike camera-based systems, it operates effectively in low-visibility conditions, making it ideal for subsea applications. Roman and Singh's work *"A Self-Consistent Bathymetric Mapping Algorithm"* [7] presents a sonar-based SLAM method that generates 3D bathymetric maps—dense point clouds of the seafloor—without external navigation aids like GPS or Long Baseline (LBL) systems. Instead, it corrects navigation drift by aligning sonar scans across multiple passes.
The algorithm adopts a submap-based SLAM framework, where overlapping submaps are locally registered and globally refined. Pose estimation is handled via an Extended Kalman Filter (EKF), which fuses constraints between submaps to improve localization.
A key innovation is reducing navigation error, the dominant source of inaccuracy in deep-sea mapping. The system performs terrain matching in two stages: a 2D correlation-based alignment followed by Iterative Closest Point (ICP) refinement. This improves localization by matching overlapping terrain features across scans.
Validated on real-world expeditions, the method outperforms dead-reckoning and LBL-based approaches, achieving map accuracy close to the sonar's resolution. However, the system depends on reasonable initial localization and may struggle in featureless terrain where scan alignment becomes unreliable.

2-3 Sensor Technologies for Underwater Navigation

This section outlines the main sensor technologies used in this thesis for underwater navigation.
Cameras are essential in visual SLAM systems but face challenges underwater due to light scattering, absorption, and low contrast, leading to noisy images [8], [9]. The environment often lacks distinct visual features, making feature detection and matching difficult [10].
Sonar is ideal for low-visibility environments. In this thesis, Side-Scan Sonar (SSS) is used for SLAM, capturing sonar images to map the seafloor and estimate the vehicle's trajectory. An example of the Side-Scan Sonar (SSS) and how it is taken is shown if figure 2-1



(a) Side-scan sonar data collection [11].



(b) Post-processed side-scan sonar image [12].

Figure 2-1: Side-scan sonar operation and output.

Inertial Measurement Units (IMUs) measure acceleration, angular velocity, and magnetic heading. They provide short-term motion estimates but suffer from drift over time. Accuracy improves when combined with other sensors, such as DVLs.
Doppler Velocity Logger (DVL) measure velocity relative to the seafloor or water column using acoustic Doppler shifts. They offer low-drift, high-accuracy velocity estimation and support both Bottom-Lock and Water-Track modes [13], [14].

2-4 The SDP-GD Framework

The paper *"Linear Time-Varying Parameter Estimation: Maximum A Posteriori Approach via Semidefinite Programming."* [1] by Vakili, Sasan, et al. published in IEEE Control Systems Letters (2023), proposes the semidefinite programming - gradient descent (SDGP-GD) algorithm. This algorithm is designed to estimate the unknown time-varying output map (C_k) of a linear time-varying system. The method is based on a Maximum A Posteriori (MAP) estimation framework, which applies Bayesian principles to incorporate prior information into the estimation process.

To solve the MAP estimation problem, the authors combine convex optimization and gradient-based refinement techniques. The initial estimate is obtained using a lifted model representation and solved through semidefinite programming (SDP) via linear matrix inequalities (LMIs). This approach leverages the computational efficiency of convex optimization while preserving the flexibility to refine the solution using gradient-based methods. Together, these techniques provide a robust and efficient framework for estimating time-varying output maps in linear dynamical systems [1].

According to the authors, this method outperforms traditional techniques such as Expectation Maximization (EM) and the Dual Kalman Smoother (DKS), offering improved accuracy and reduced computational complexity.

This section gives a quick summary about the problem formulation, afterwards the lifted model representation is introduced, followed by an overview of the MAP estimation framework and then discusses the conservative LMI approximation.

Problem Formulation

A discrete-time, linear, time-varying system defined by a process model and a measurement model is used. These have rthe following form:

Process Model:

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \quad k \in \mathbb{Z}_+ \tag{2-1}$$

- x_k: state vector, u_k: input vector
- A_k: state transition matrix, B_k: input matrix
- $w_k \sim \mathcal{N}(0, \sigma_{w_k})$: process noise

Measurement Model:

$$y_k = C_k x_k + v_k, \quad k \in \mathbb{Z}_+ \tag{2-2}$$

- y_k: output measurement, C_k: unknown observation matrix
- $v_k \sim \mathcal{N}(0, \sigma_{v_k})$: measurement noise

Output Map Dynamics: The observation matrix C_k is parameterized by a time-varying vector θ_k. The goal is to estimate the sequence $\{\theta_k\}$ over time, enabling reconstruction of the evolving measurement model.

$$\theta_{k+1} = \theta_k + \eta_k, \quad \eta_k \sim \mathcal{N}(\mu_{\eta_k}, \sigma_{\eta_k}) \tag{2-3}$$

Lifted Model Representation

To estimate the unknown parameter vector θ in the output map $C(\theta)$, the SDP-GD algorithm is applied using input-output measurement data. The output is given by:

$$y = C(\theta)Au = C(\theta)X_{\text{nominal}}$$

where u is the lifted input vector defined as:

$$u = \left[\mu_{x_0}^\top, (B_0 u_0)^\top, \ldots, (B_{n_\tau - 1} u_{n_\tau - 1})^\top\right]^\top$$

This lifting follows the formulation introduced in the SDP-GD framework to represent the full trajectory compactly over a time horizon.

The lifted model reformulates a time-varying linear system over an entire time horizon as a single large-scale static system. Instead of estimating parameters at each time step separately, all time-varying observation matrices C_k are collected into a single high-dimensional parameter vector θ [1]. This enables global optimization using all available data through a maximum a posteriori (MAP) estimation framework.

The lifted model consists of four components:

- **Lifted Process Model:**

  $$x = A(u + w_x)$$

  where x is the stacked state vector, w_x the process noise, and A a lower-triangular matrix encoding system dynamics across time.

  Here u is equal to

  $$u = \left[\mu_{x_0}^\top, (B_0 u_0)^\top, \ldots, (B_{n_\tau - 1} u_{n_\tau - 1})^\top\right]^\top \tag{2-4}$$

- **Lifted Observation Model:**

$$y = C(\theta)Au + w_y(\theta), \quad \text{with} \quad w_y(\theta) = C(\theta)Aw_x + v$$

  where y is the stacked measurement vector, $C(\theta)$ a block-diagonal matrix of parameter-dependent observations, and v the measurement noise.

- **Lifted Parameter Dynamics:**

$$\theta = \mu_\theta + w_\theta, \quad w_\theta = D\eta$$

  where μ_θ is the prior mean and D encodes parameter evolution.

- **Noise Covariance:**

$$\begin{aligned} w_x &\sim \mathcal{N}(0, \Sigma_{w_x}), \quad \Sigma_{w_x} = \text{diag}(\Sigma_{x_0}, \dots, \Sigma_{w_{n_T-1}}) \\ v &\sim \mathcal{N}(0, \Sigma_v), \quad \Sigma_v = \text{diag}(\Sigma_{v_0}, \dots, \Sigma_{v_{n_T}}) \\ w_\theta &\sim \mathcal{N}(0, \Sigma_{w_\theta}), \quad \Sigma_{w_\theta} = D\Sigma_\eta D^\top \end{aligned}$$

This lifted formulation allows for efficient and consistent parameter estimation over the full trajectory by leveraging temporal dependencies in both state and measurement models.

MAP Estimation Framework

The parameter estimation problem is approached using a Bayesian framework, which incorporates prior knowledge to improve the accuracy of the estimates. Specifically, the algorithm maximizes the posterior distribution of the parameters θ given the observed data y and inputs u:

$$\hat{\theta} = \arg\max_\theta \; p(\theta \mid y, u) \tag{2-5}$$

This is equivalent to minimizing the negative log-posterior, which leads to the following MAP cost function:

$$\begin{aligned} J(\theta) = &\log\det\left(\Sigma_{wy}(\theta)\right) + \\ &\|y - C(\theta)Au\|^2_{\Sigma_{wy}(\theta)^{-1}} + \|\theta - \mu_\theta\|^2_{\Sigma^{-1}_{w\theta}} \end{aligned} \tag{2-6}$$

Here, $\Sigma_{wy}(\theta)$ is the covariance matrix of the combined process and measurement noise. The matrix $C(\theta)$ represents the observation model as a function of the parameter vector θ, and A, u, and y correspond to the system dynamics matrix, control inputs, and measured outputs, respectively. The term μ_θ denotes the prior mean of the parameter vector, and $\Sigma_{w\theta}$ is the associated prior covariance.

The goal is to find the parameter vector that minimizes this cost function:

$$\theta^* = \arg\min_\theta J(\theta)$$

LMI Conservative Approximation

The cost function defined in eq. 2-6 is a non-convex problem and thus difficult to optimize directly. To address this, a convex approximation is constructed using Linear Matrix Inequalities (LMIs), allowing the MAP estimation problem to be solved efficiently within a semidefinite programming (SDP) framework. The LMI-based conservative formulation has the following structure [1]:

$$
\begin{aligned}
\min_{S,\theta,\gamma,\beta} \quad & \operatorname{tr}(S - I) + \gamma + \beta \\
\text{s.t.} \quad & \begin{bmatrix} -\Sigma_{w_x}^{-1} & A^\top C(\theta)^\top \\ C(\theta)A & \Sigma_v - S \end{bmatrix} \succeq 0, \\
& \begin{bmatrix} -S & y - C(\theta)Au \\ (y - C(\theta)Au)^\top & -\gamma \end{bmatrix} \succeq 0, \\
& \begin{bmatrix} -\Sigma_{w_\theta} & \theta - \mu_\theta \\ (\theta - \mu_\theta)^\top & -\beta \end{bmatrix} \succeq 0,
\end{aligned}
\tag{2-7}
$$

where S, θ, γ, β are optimization variables.

This LMI-based approximation provides an estimate for θ, which serves as a "warm start" for further optimization. To refine this initial estimate, a gradient-based method is employed, namely the BFGS algorithm. This algorithm is a quasi-Newton method that is used to minimize the original non-convex cost function defined in eq. 2-6. This second-stage refinement improves the accuracy of the solution and benefits from a superlinear convergence rate [1].

Chapter 3

System Setup and Experimental Platform

This chapter outlines the system architecture and experimental setup used to evaluate the performance of the SDP-GD in a simulated underwater environment. The simulation is developed using the Gazebo simulator, integrated with ROS for control, visualization, and data handling. Visual and sonar-based SLAM configurations are implemented, and the Desistek SAGA and the BlueROV2 underwater vehicle model is used as the experimental platform. The implementation of the BlueROV2 using the ORB-SLAM2 algorithm is available in the GitHub repository [15], under the project name *Orca4*, while the implementation for the Desistek SAGA ROV can be found in the GitHub repository [16]. The initial plan was to implement both algorithms on the BlueROV2. However, due to limitations in the Orca4 project and ROS 2 framework, the Desistek SAGA was implemented using the ROS (ROS 1) framework instead. The chapter also describes environmental factors affecting the experiments, as well as the approach used to obtain and utilize ground truth data.

3-1 Overview of the Gazebo Simulation Environment

The simulation environment is built using the Gazebo simulator and integrates the BlueROV2 model from Blue Robotics [17]. This model represents the autonomous underwater vehicle (AUV) used in the experiments and is linked to Gazebo for conducting underwater testing.

In the simulation, it is assumed that the BlueROV2 possesses hydrodynamic symmetry. This means the vehicle is considered to not exhibit complex hydrodynamic interactions or coupling effects, such as drag, lift, and changes in orientation due to fluid dynamics. By minimizing these effects, the vehicle's behavior becomes more stable and predictable, thereby simplifying the control and navigation processes within the simulator.

3-2 Utilities

For the simulation and computation of the output maps, the following laptop was used:

Table 3-1: Specification of the laptop/processor used in this thesis [18]

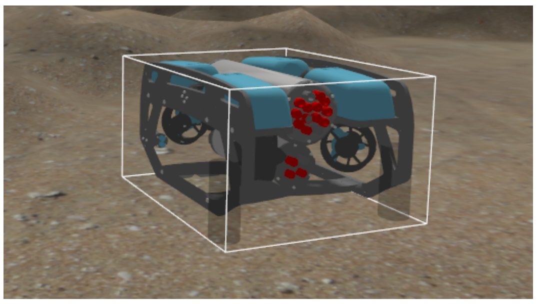
Model name	HP ZBook Power 15.6 inch G9 Mobile Workstation
Processor	Intel Core i7-12700H Processor, 2.3 GHz base frequency, up to 4.7 GHz with Intel Turbo Boost Technology, 24 MB L3 cache, 14 cores, and 20 threads
RAM	16 GB DDR5-4800 MHz
GPU	Integrated Intel Iris Xe (optionally: NVIDIA RTX A1000 Graphics Card, 4 GB GDDR6 dedicated)

3-3 Visual SLAM Configuration

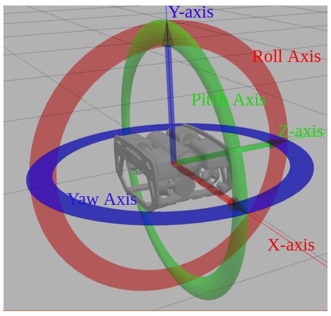
Hardware Components: Orca and BlueROV2

The BlueROV2 from BlueRobotics is used as the AUV for simulation. It comes in a base model and a heavy configuration (details in Appendix A-2). For this research, the base configuration is selected, as it is sufficient for maintaining a constant height in calm environments. Key parameters are listed in Appendix Table A-1.

Figure 3-1 shows the BlueROV2 in the Gazebo simulator, along with its axis orientation (**note: the Y- and Z-axes should be switched**).



(a) BlueROV2 in Gazebo

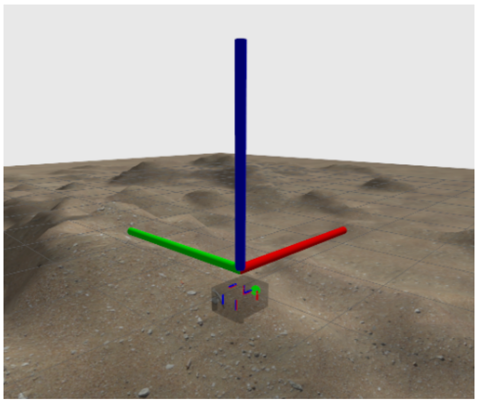


(b) AUV axis orientation (Y- and Z-axes switched)

Figure 3-1: BlueROV2 model and axes in the Gazebo environment.

Underwater dynamics are simulated using three Gazebo plugins: BuoyancyPlugin, HydrodynamicsPlugin, and ThrusterPlugin, modeling buoyancy, hydrodynamic forces, and propulsion respectively. Details on these plugins and coordinate frames are provided in Appendix A-1.

The Orca4 project [15] integrates the BlueROV2 with ORB-SLAM2 inside Gazebo, enabling realistic underwater SLAM simulations. An example output from Orca4 is shown in Figures 3-2.


(a) BlueRov2 in its environment, taken from [15]

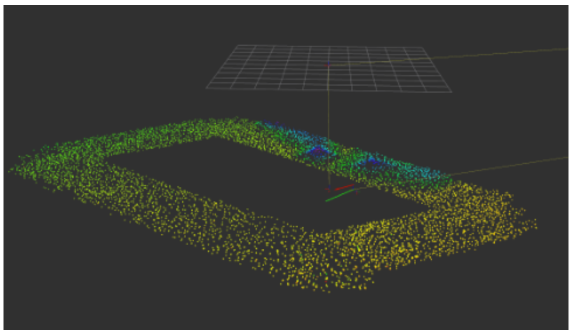

(b) Map of the environment using ORB-SLAM 2.0, taken from [15]

Figure 3-2: BlueROV2 and its SLAM-generated environment map

In addition to ORB-SLAM2, the Orca4 system integrates several key modules: BaseController, ArduPilot, MAVROS, Nav2, and the ThrusterPlugin. The BlueROV2 operates within the ROS2 ecosystem, using the MAVLink protocol via ArduSub. MAVROS acts as the communication bridge between ArduSub and ROS2.

The system features a tightly integrated control and estimation pipeline: **ORB-SLAM2**: Estimates the camera pose using stereo images, with loop closure and optimization to reduce drift and ensure global map consistency.
ArduPilotPlugin: Simulates IMU and barometer readings by adding noise to the ground truth pose (noise parameters not specified).
ArduSub: Fuses vision-based SLAM output with simulated sensor data using a Kalman filter to refine state estimation.
Controller: Computes control commands in x, y, z, and yaw directions based on the estimated state and sends them to the
ThrusterPlugin, which simulates propulsion dynamics.

SLAM Algorithm: ORB-SLAM2

The SLAM algorithm used in this configuration is ORB-SLAM2 [19], which supports both stereo and RGB-D input. It uses depth information to define a stereo coordinate system for extracted features, enabling it to function independently of the input type.
A schematic of the ORB-SLAM2 system is shown in Figure 3-3, and a brief overview is provided below.

The system maintains two key graph structures:

- A **covisibility graph**, connecting keyframes with shared map points.
- A **spanning tree**, ensuring global map consistency and efficient optimization.

These graphs help localize the camera within relevant keyframe neighborhoods and enable scalable operation across large environments. Pose-graph optimization during loop closure ensures long-term accuracy.

Main Components

ORB-SLAM2 consists of three main modules:

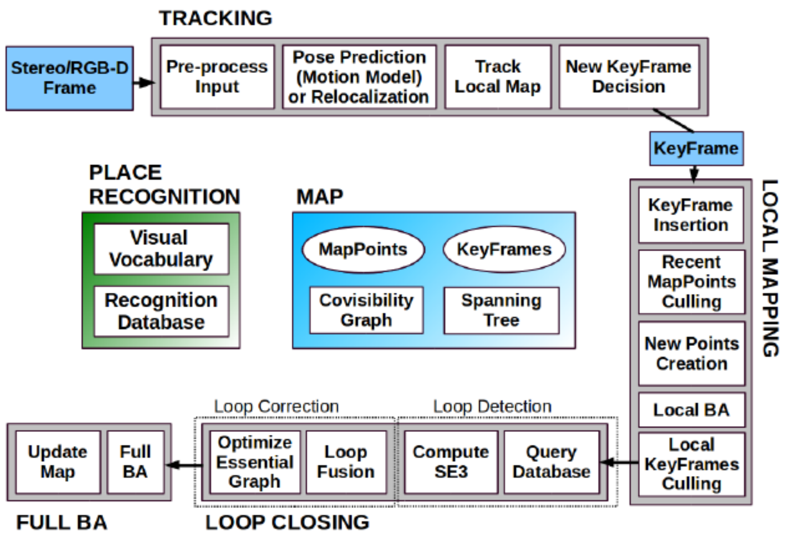


Figure 3-3: ORB-SLAM2 architecture [19]

- **Tracking:** Estimates the camera pose by matching features in the current frame with the local map. Pose refinement is performed using motion-only bundle adjustment.
- **Local Mapping:** Maintains and refines a local map through local bundle adjustment over recent keyframes and associated map points.
- **Loop Closing:** Detects revisited locations to correct accumulated drift by aligning current observations with earlier map regions. Loop closure is not used in this thesis, as the trajectory is linear with no revisits (see Appendix A-3).

Details on system initialization and keypoint detection are provided in Appendix A-3.

Visual Map Comparison and Offline Map: Fig. 3-4 illustrates a comparison between the mapped datapoints and the simulated environment. The map accurately represents the two hills in the image, marked with blue and purple dots indicating different heights.

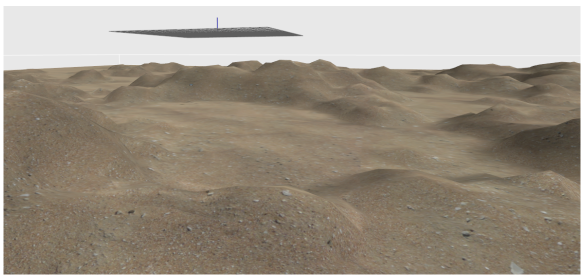
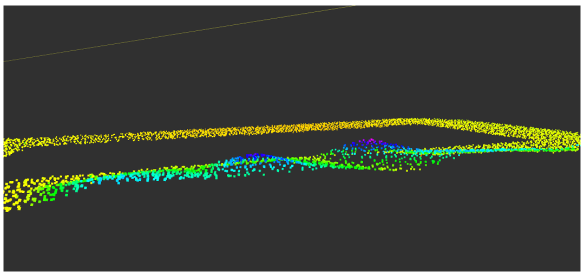


Figure 3-5: Small Hill identified in the simulator



The constructed map is represented as a point cloud, which serves as the basis for comparing the SDP-GD and ORB-SLAM2 algorithms. While the SDP-GD is not originally designed for offline operation, using the current run's estimated positions in with the map built during a previous run allows for a consistent evaluation of its performance.

The k-nearest neighbors (KNN, see Appendix B-1 for more info on KNN) is applied between the current trajectory and the prebuilt map to reconstruct the seabed. This method provides a fair comparison because both algorithms are assessed using the same reference environment, and the evaluation focuses on how accurately each method can align to and reconstruct that environment. Moreover, using the same point cloud ensures that differences in performance stem from the SLAM algorithm itself, not from variations in the input data or mapping process.

Seabed Complexity: The seabed complexity in this experiment is not varied as done for the Sonar-Based SLAM evaluation. The complexity is altered by scaling the seabed along the Z-axis, resulting in deeper hills and valleys. For this evaluation, only two different scaling factors are applied: 1 and 3. Due to lack of time, the differences and specification of the different seabed couldn't be provided.

Performance Evaluation

ORB-SLAM2 is designed to be lightweight and computationally efficient, making it suitable for deployment on systems with limited processing capabilities, such as those with standard consumer-grade CPUs. Its performance has been tested on various datasets, including KITTI, EuRoC, and TUM RGB-D. In these evaluations, ORB-SLAM2 has been compared to other state-of-the-art stereo SLAM systems, such as LSD-SLAM. The results demonstrate that ORB-SLAM2 achieves significantly better performance and is an effective algorithm. Drifts can still occur in the simulator, therefore experiments are performed to determine the amount of drift of position estimation.

Drift: In this experiment, the AUV remains stationary for a period to measure how far it drifts with the controller turned off. The experiment was conducted at two positions: $(x, y, z) = (0, 0, -2.5)$ (near the surface) and $(x, y, z) = (0, 0, -6.5)$, as illustrated in figure 3-6. After 480 seconds, the AUV at -2.5 meters had moved to: $(\Delta x, \Delta y, \Delta z) = (0, 0.470, -1.74)$, and at -6.5 meters to $(\Delta x, \Delta y, \Delta z) = (0, -480, -2.133)$ in a nonlinear manner.

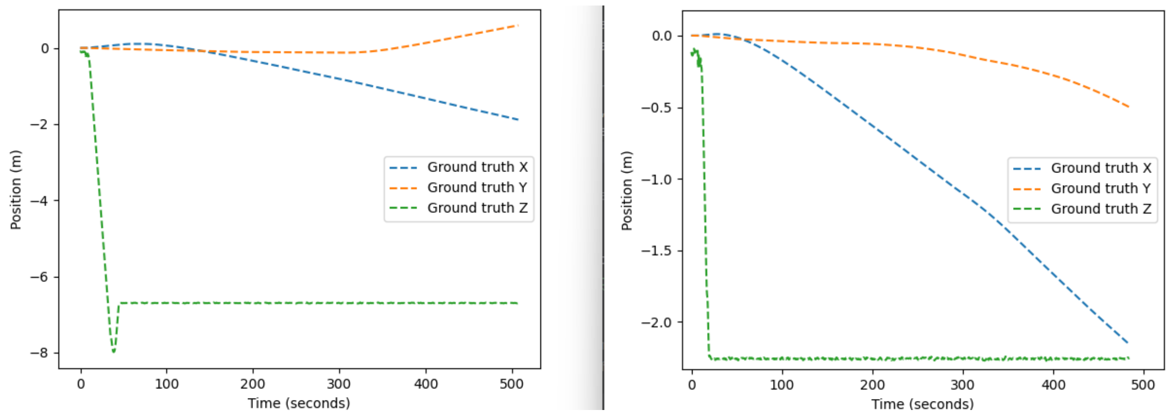
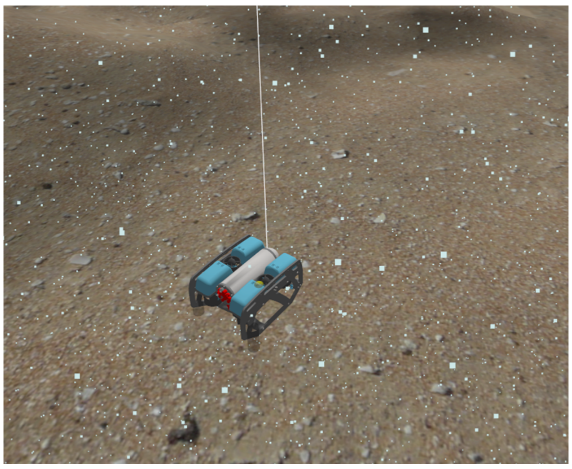


Figure 3-6: Drift over time while standing still, due to water current. On the left, the AUV is at $(x, y, z) = (0, 0, -2.5)$; on the right, it is at $(x, y, z) = (0, 0, -6.5)$

It can be observed that the AUV is not neutrally buoyant without the controller, as it sinks and reaches the bottom within 20 seconds. In contrast, the AUV exhibits a slow drift in the horizontal plane, moving approximately 2.5 meters in the x direction and 0.5 meters in the y direction over a period of 500 seconds. This indicates that the simulator includes a low-level current, although its exact specification could not be determined due to the nested structure of the simulation code. However, with the controller enabled, the AUV remained stationary


(a) Simulator environment with visual noise (white particles).

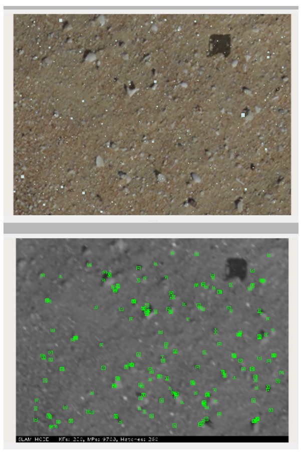

(b) Top: raw camera image; bottom: filtered murky view.

Figure 3-7: Visual noise simulation in Gazebo.

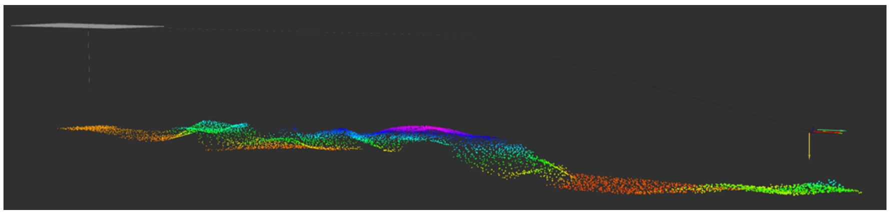


Figure 3-8: Point cloud in clear water conditions (ideal case).

throughout the entire duration, demonstrating high precision and stability in maintaining a fixed position.

Adding Noise

To simulate realistic underwater conditions, visual noise is introduced in the form of suspended particles, creating murky water. As shown in Figures 3-7a and 3-7b, 15,000 particles/m^2 are added, moving randomly at 0.01 m/s. A gray filter is applied to mimic underwater light absorption, affecting the ROV's camera view.

In clear conditions, the AUV generates accurate point clouds while moving straight (Figure 3-8). However, with noise added, the point cloud deviates significantly (Figure 3-9). Although the added visual noise is mild, ORB-SLAM2 is still misled due to the AUV's proximity to the seabed. Suspended particles are falsely detected as keypoints and persist until leaving the field of view, gradually distorting the map.

This highlights ORB-SLAM2's limitations in noisy underwater environments. To address this, a sonar-based SLAM approach is used, which is inherently more robust in low-visibility conditions due to its reliance on acoustic signals rather than visual features.

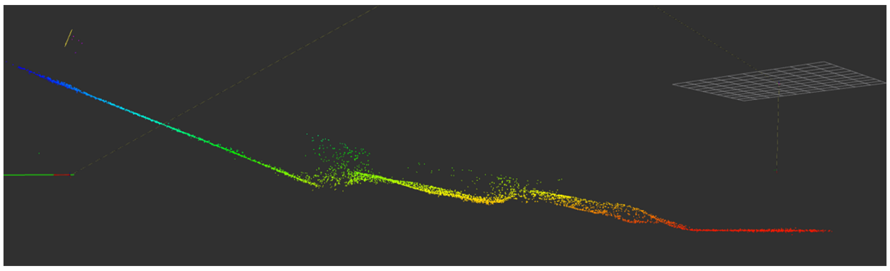


Figure 3-9: Point cloud under murky conditions, showing deviation from ground truth.

3-4 Sonar Based SLAM Configuration

Hardware Components: UUV Simulator

The paper "UUV Simulator: A Gazebo-Based Package for Underwater Intervention and Multi-Robot Simulation" [20] provides an open-source underwater robotics simulation framework built for the the Gazebo simulator. This package aims to address challenges of unmanned underwater vehicle (UUV) simulation, including hydrodynamic modeling, sensor integration, and multi-robot coordination. The UUV Simulator provides a high-fidelity and modular simulation environment which can be used for advanced underwater robotics applications/simulation.

Key Contributions

- Hydrodynamic and hydrostatic modeling to simulate underwater forces and also thruster dynamics, robotic manipulators, and environmental disturbances
- Provides modular vehicle configurations for different mission requirements.
- Includes acces to Robot Operating System (ROS) for real-world deployment.
- Includes sensor such as Doppler Velocity Loggers (DVLs), pressure sensors, and sonar for various applications.
- Includes a multi-beam echo sounders for bathymetric mapping.
- Supports underwater light attenuation, particle effects, and water currents.

Desistek SAGA ROV

The Desistek SAGA ROV Simulator is an open-source package designed to simulate the real Desistek SAGA Remotely Operated Vehicle (ROV) (see Fig. 3-10) within the Unmanned Underwater Vehicle (UUV) Simulator framework. The package comes with a robot description in the form of URDF/Xacro files, which detail the SAGA ROV's physical and dynamic characteristics, ensuring accurate simulation of its behavior underwater.

The performance of the Desistek SAGA ROV is similar to that of the ORB-SLAM2-based setup, as both utilize the same plugins. However, the Desistek SAGA ROV offers more flexibility, including the ability to simulate variable water currents and more realistic visibility noise.

The ROV is equipped with three thrusters: one oriented along the z-axis and two along the x-axis. To achieve motion in the y-axis, the vehicle must first rotate and then apply thrust in the desired direction. This ROV is particularly suitable for this project because it simplifies controller design. Specifically, the controller can focus on maintaining constant velocity along the z-axis while primarily maneuvering in the x-direction.

Figure 3-10: Image of the real Desistek SAGA ROV used in simulation. The model is sourced from [21].

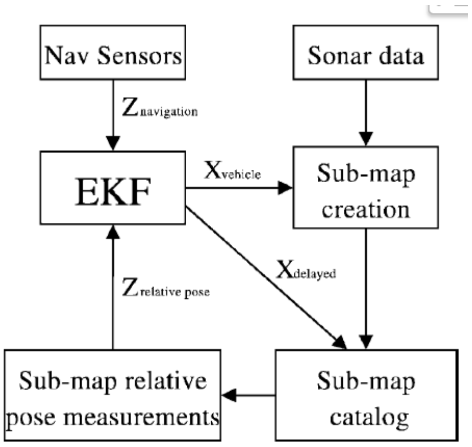


Figure 3-11: Block diagram of the delayed-state EKF. The algorithm uses navigation data to generate small, localized bathymetric submaps. The origins of these submaps are stored in a delayed state vector, which enables the computation of relative pose constraints to correct navigation drift. Image taken from [7]

SLAM Algorithm: Sonar Based SLAM

The implemented algorithm is based on the paper *"A Self-Consistent Bathymetric Mapping Algorithm" by Chris Roman and Hanumant Singh* [7] which introduces a SLAM approach for underwater mapping using sonar data.
The algorithm follows a submap-based SLAM approach, where the seafloor is mapped in overlapping segments (submaps). These are locally aligned and globally refined to improve consistency. Vehicle pose is estimated using an Extended Kalman Filter (EKF), which incorporates relative constraints between submaps based on terrain matching.

Terrain alignment is performed in two stages: a correlation-based 2D match, followed by refinement using k-nearest neighbors (KNN). This method improves map accuracy by correcting navigation errors, though it depends on reasonably accurate dead-reckoning and performs best in terrain with distinguishable features.

A schematic overview of the algorithm is shown in Figure 3-11.

In this thesis, the pose estimation method from the Sonar-Based SLAM algorithm is adapted for a single submap. The SDP-GD requires a linear system model is used, and therefore a standard Kalman Filter is employed instead of an Extended Kalman Filter. Due to time constraints, the full submap-based approach was not implemented.

One key comparison is to determine how far the SDP-GD algorithm can track before error growth becomes significant. This distance can inform the optimal size of a submap for future implementations.

The pose of the system is estimated using measurements from the DVL and IMU, combined through a rotational transformation matrix.

Transformation from Body Frame Velocity to Inertial Frame Velocity

The conversion from body-frame velocity to the inertial) frame is achieved using a nonlinear rotation matrix derived from the vehicle's attitude (Euler angles).

The transformation is expressed as:

$$v_l = R(\phi, \theta, \psi) \cdot v_b \tag{3-1}$$

Where:

- v_b is the velocity vector in the body frame
- v_l is the velocity vector in the inertial frame
- $R(\phi, \theta, \psi)$ is the rotation matrix constructed from the vehicle's *roll* ϕ, *pitch* θ, and *yaw (heading)* ψ

In the state-space model in the paper has the following form:

$$\dot{x}_v(t) = f(x_v(t)) + w(t) \tag{3-2}$$

The state space model is adopted to also include the input into the system, in order to have a fair comparison between the SDP-GD and the Sonar Based SLAM. The positional components are updated by:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R(\phi, \theta, \psi) \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3-3}$$

Where:

- u, v, w are linear velocities in the body frame
- x, y, z are positions in the local-level frame and also the state with the velocities in the inertial frame for these axes (v_x, v_y, v_z).

The standard aerospace XYZ Euler angle rotation matrix from body to inertial frame is:

$$R(\phi, \theta, \psi) = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{3-4}$$

The orientation angles are obtained from the IMU sensor, while the body-frame velocities u, v, w are measured by the DVL sensor. It is assumed that the IMU provides angles in the inertial frame. The specific IMU used in this setup is the ADIS16448 (see the appendix for more details A-4). However, the datasheet for this sensor does not explicitly state the reference frame for the orientation output.

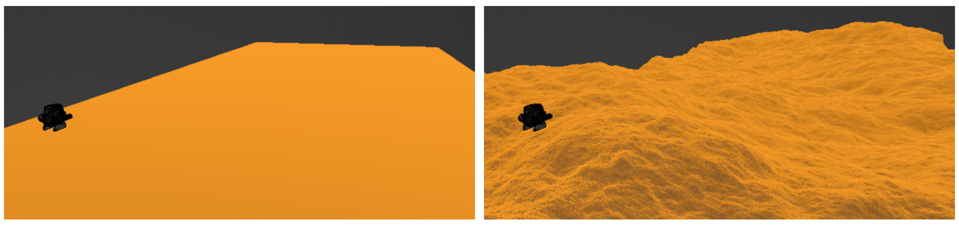
Offline Map

The SDP-GD algorithm is executed offline, and for a fair comparison, the map generated by the Sonar-Based SLAM is also utilized in an offline setting. Using a system model that incorporates the control input, the states—specifically the position of the ROV—are computed at each timestep. These position estimates $((\hat{x}, \hat{y}, \hat{z}))$ are then refined using a Kalman Filter to improve accuracy. Finally, KNN is applied to align the current estimate with the previously generated map from an earlier run, this generating the seabed reconstruction.

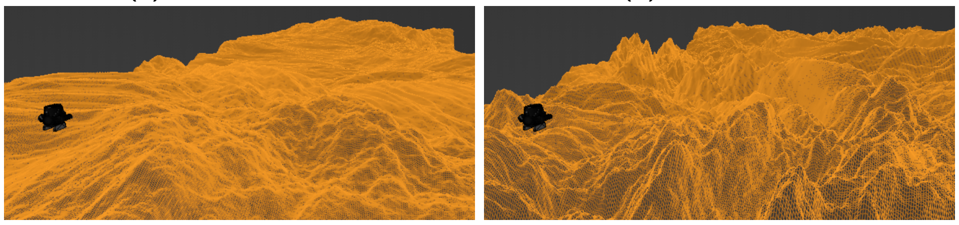
3-5 Environmental Considerations

Seabed Complexity and Terrain Characteristics

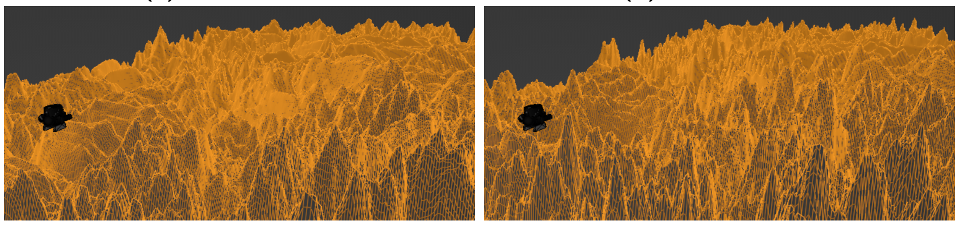
The seabed is design using Blender, a free open-source 3D computer graphics software. The surfaces are made starting off with a flat surface. By adding *fractal Perlin nois* the patterens of the map is created. The parameter used to create different map complexity are scale and distortion. The **Scale** parameter in the Noise Texture node controls the frequency of the noise pattern. The **Distortion** parameter introduces a warping effect to the noise pattern. Increasing the distortion value causes the noise to appear more twisted and irregular, which can be useful for creating more complex and less uniform textures.



(a) Initial flat surface **(b)** Distortion = 0



(c) Distortion = 1 **(d)** Distortion = 4



(e) Distortion = 8 **(f)** Distortion = 12

Figure 3-12: Generated seabed terrain using Fractal Perlin noise with increasing distortion levels. The initial flat surface is progressively transformed into more complex and irregular terrains, mimicking underwater environments ranging from simple sandbeds to chaotic cave-like structures.

Typical Underwater Current Velocities

Current underwater has usually an average velocity and variation. In the simulation the speed is set, based on real world underwater currents. Typical underwater current velocities vary, depending on environment and specific oceanographic conditions. Various underwater current velocities are [22]:

- **Deep-sea environments:** Thermohaline circulation with currents between 2 cm/s and 20 cm/s.
- **Benthic storms:** Underwater velocities reach up to 73 cm/s.
- **Strait of Gibraltar:** Maximum flow velocities of 300 cm/s.
- **Coastal regions and tidal inlets:** Maximum current speeds 1 m/s.
- **Straits and estuaries:** Tidal currents can reach speeds as high as 3 m/s.

Chapter 4

Controller Design

This chapter focuses on the design, tuning, and evaluation of a Proportional-Integral-Derivative (PID) controller for two underwater vehicles: the Desistek SAGA AUV and the BlueROV2.

The first part of the chapter presents the structure and implementation of the PID controller developed for the Desistek SAGA, along with gain selection strategies and performance evaluation under various simulated inputs. The controller's robustness is then tested under external disturbances, such as varying current velocities and directions.

The latter part of the chapter discusses the challenges encountered in implementing a similar controller for the BlueROV2, and the decision to instead use the built-in Orca controller for experiments involving visual SLAM. Additional experiments are conducted to assess the straight-line motion accuracy, error accumulation during repeated traversals, and the influence of measurement noise on pose estimation.

4-1 PID Controller Design

The system operates in a closed-loop configuration, where the controller continuously adjusts the thruster commands based on sensor feedback to minimize tracking error. The controller design is a discrete PID controller.

This section first introduces the controller structure and its implementation, including the interaction between reference inputs, feedback, and thruster commands. It then describes the design decision used to tune the PID controller gains and ensure desirable performance , i.e. low overshoot, stability, and robustness.

Controller Structure and Implementation

The control system takes as reference input the desired velocity $\eta_{p_{ref}}$, and uses as feedback the estimated current velocity $\hat{\eta_p}$ obtained from the DVL and IMU sensor fusion. The input to the PID controller is the velocity error:

$$e[k] = \eta_{p_{ref}}[k] - \hat{\eta_p}[k] \tag{4-1}$$

To evaluate system stability and controller response, the AUV is subjected to a sequence of time-varying reference velocities along the x, y, and z axes. The reference signal used in simulation is:

$$u_i = \begin{Bmatrix} 0.01 & 0.10 & 0.20 & 0.50 & 0.70 & 0.10 & 1.20 \end{Bmatrix}$$

Table 4-1: Step Response Data (Raw Sensor Data)

Input amplitude x velocity	0.01	0.1	0.2	0.5	0.7	1.0	1.2
Rise Time	0.64	0.40	0.68	0.31	0.61	0.82	0.71
Transient Time	9.50	9.88	6.31	9.86	9.78	9.50	9.69
Settling Time	9.50	9.88	6.31	9.86	9.78	9.50	9.69
Settling Min	0.81	0.74	0.92	0.60	0.92	0.78	0.82
Settling Max	1.02	1.02	1.03	1.02	1.01	1.00	1.04
Overshoot (%)	3.63	30.54	1.81	60.01	3.31	0.58	3.20
Undershoot (%)	0.05	0.02	0.01	0.01	0.00	0.01	0.00
Peak	1.02	1.02	1.03	1.02	1.01	1.00	1.04
Peak Time	7.60	7.30	8.20	8.50	7.90	7.90	7.60

The computed control signal $u[k]$ is then passed into the thruster plugin in the simulation environment, which translates it into rotational angular velocity (ω) commands for the individual thrusters. These commands produce the necessary thrust forces that move the AUV and drive it toward the desired velocity.

PID Gain Selection

The ROV will mainly propagate in a straight line in the x-direction. Therefore, it should not rotate in any other direction and should maintain its orientation around the z-axis. The ROV does not have the ability to move directly in the y-direction (left and right), as this would require rotation, due to not having a thruster that directly thrust in the y direction.

The PID controller gains are selected to achieve the following characteristics:

- **Rise Time:** A rise time that is too fast can lead to overshoot, which in turn can cause oscillations.
- **Transient Time:** The transient time in the x-direction is not critical due to the requirement to go straight. However, it should preferably not exceed the total simulation time. In the other directions, transient time is more important to prevent drift.
- **Low Overshoot:** Overshoot should be minimized to avoid oscillations, especially given the high resistance underwater.

4-2 Performance Analysis of the PID Controller

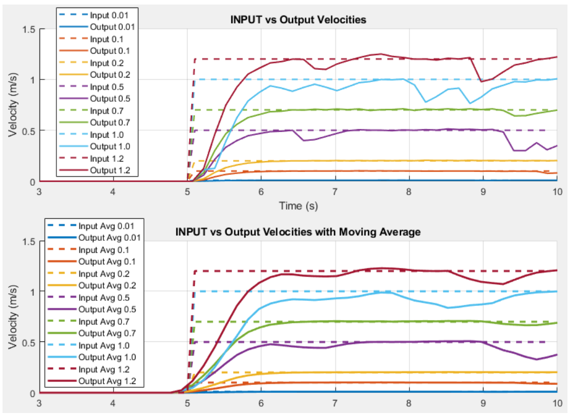
Step Response Evaluation

The PID controller's performance is assessed using the system's step response in the x-direction. Velocity data is obtained from the DVL and IMU, both of which introduce sensor noise.
Table 4-1 presents the step response specifications based on raw (noisy) measurements.

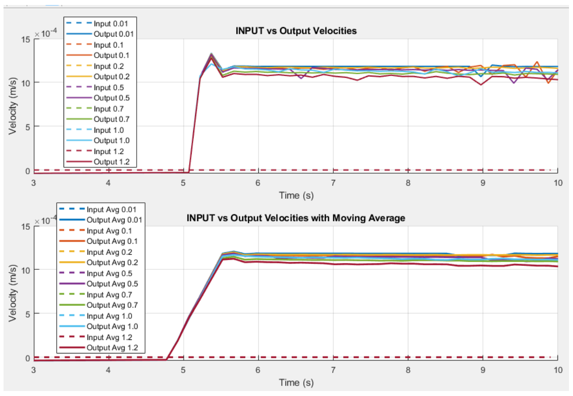
At lower input amplitudes (0.1 and 0.5), the controller exhibits considerable overshoot—30.54% and 60.01%, respectively. In contrast, higher input amplitudes (1.0 and 1.2) show significantly reduced overshoot, indicating improved stability at those levels.
Transient and settling times remain consistent across input amplitudes, demonstrating reliable transient behavior. The settling maximum hovers around 1.0, reflecting uniform steady-state performance. No significant undershoot is observed, and peak times are relatively stable.
Based on these findings, an input amplitude of 1.0 is selected for all subsequent experiments in this thesis, as it offers a good balance of low overshoot and reliable steady-state response.

Directional Response Analysis

The following figures show the velocity step response in each axis.



(a) Enter caption for X-axis step response



(b) Enter caption for Y-axis step response

Figure 4-1: Comparison of step responses along X and Y axes

Note that the output response in the y-direction does not follow the input function. This is because the ROV cannot move directly in the y-direction without rotating. While this is beneficial when the goal is to move only in the x-direction, it can become a disadvantage in environments with lateral disturbances, such as water currents hitting the side of the ROV.

Z-Direction

The velocity in the z-direction was kept at zero amplitude throughout the experiment. As shown in Figure 4-4, the blue signal represents the system's response, which slowly deviates by approximately 0.1 meters over a period of 120 seconds.

4-3 Disturbance Analysis

In this section, the ROV is simulated in water with current disturbances. These currents are modeled with both a varying velocity and direction. The current velocity η_1 ranges between 0.2 and 1.2 m/s and can act in both forward and reverse directions. Additionally, the direction of the current η_2 is modeled with a mean of 0° and a variance of 10°.

The ROV is oriented to face the direction of the current, and a constant reference velocity of 1 m/s in the x-direction is applied. The simulation results for each axis are presented and discussed below.

In appendix A-7 the current signal is plotted with the auv velocity response, but the plot of the position can give the most information on the effect of varying currents.

X-Direction

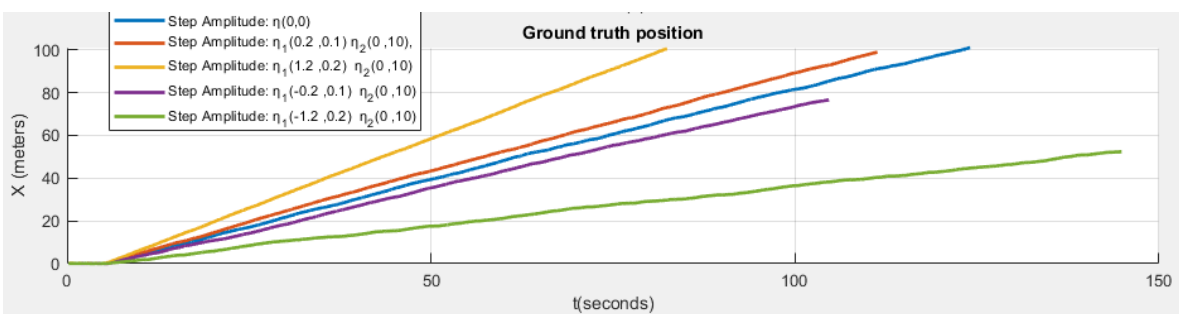


Figure 4-2: Ground truth position of the ROV under varying simulated current velocities in the x-direction.

It can be observed that varying current velocities have a linear effect on the AUV's position along the x-axis. When the current opposes the AUV's motion (e.g., the green signal with a current velocity of -1.2), it slows down the rate of position change, resulting in a reduced slope of approximately 0.4375. Whenever the current pushes from behind (e.g., the yellow signal with a current velocity of 1.2), it accelerates the AUV forward, producing a steeper slope of approximately 1.25. This indicates that the controller lacks robustness along the x-axis, as it is unable to consistently reject disturbances caused by varying current velocities.

Y-Direction

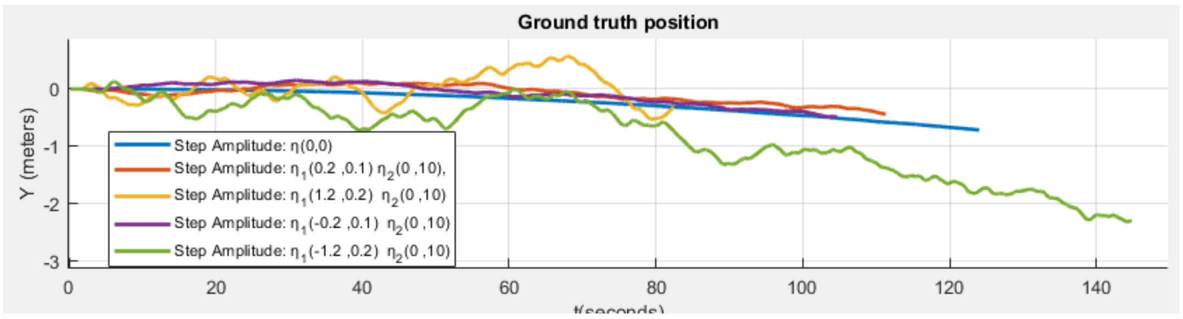


Figure 4-3: Ground truth position of the ROV under varying simulated current velocities in the y-direction.

When the ROV is facing the current, the current has little effect on the y-direction at lower velocities. However, at higher current velocities, the ROV begins to exhibit noticeable oscillations in the y-direction.

Z-Direction

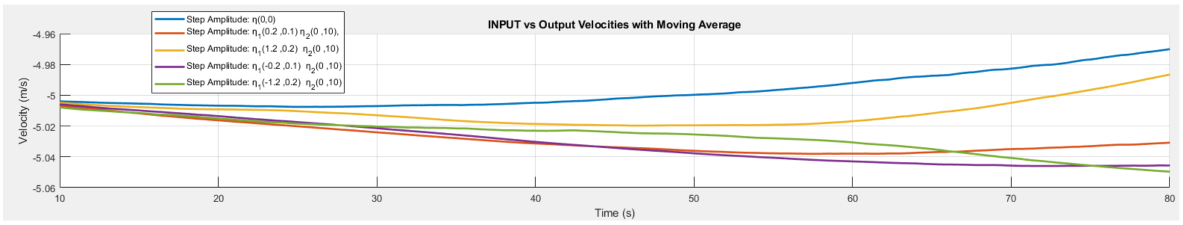


Figure 4-4: Ground truth position of the ROV under varying simulated current velocities in the z-direction.

The varying current velocity does not appear to significantly affect the ROV's z-position. This is primarily because the current does not vary in the vertical direction, only in the horizontal direction. This omission was an oversight during the implementation phase.

4-4 Controller for Visual SLAM

An initial attempt to implement a PID controller for the BlueROV2 proved significantly more difficult than for the Desistek SAGA. While the SAGA uses only three thrusters, the BlueROV2's six-thruster configuration introduces more complex control dynamics. Additional challenges included persistent noise in the simulator—difficult to tune—and environmental water currents causing lateral drift.

The feedback controller followed the same structure as that used for the SAGA: the reference input was the desired velocity, and the measured velocity from the DVL served as feedback. The best result obtained with this PID setup is shown in Figure 4-5, where the vehicle drifts into a wide circular path. This limits its use to short-range motion (e.g., under 5 meters), where deviation is minimal.

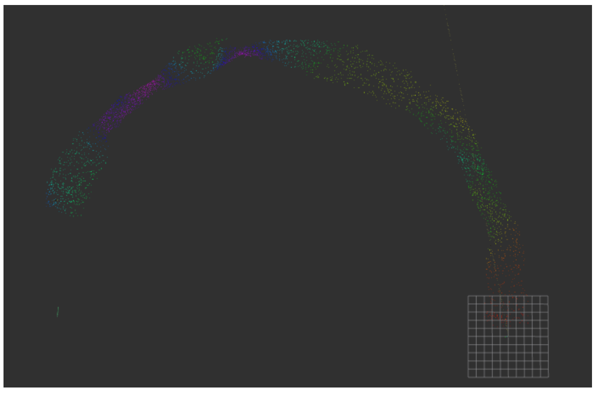


Figure 4-5: Ground truth seabed reconstruction using a PID controller.

In contrast, the built-in controller from the Orca4 package delivers significantly better performance. It uses ORB-SLAM2's estimated position for feedback and accepts reference positions as control inputs. This controller was therefore adopted for all experiments. Its main limitation is that control inputs remain fixed during each run, preventing minor real-time adjustments throughout the mission.

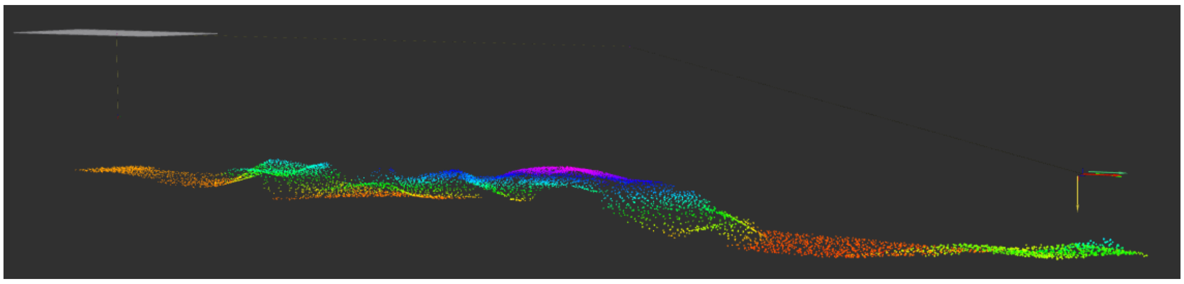


Figure 4-7: Top-down view of the map generated during an 80-meter straight-line traversal. Colors represent height variations.

Straight-Line Motion Experiments

Experiments were conducted to evaluate the AUV's ability to maintain straight-line motion at varying depths. A fixed depth of 6.5 meters was chosen to minimize collision risk while ensuring stable visual tracking over longer distances. Additional trials tested the system's ability to return to a previously visited location and assessed whether repeated passes improved pose estimation.
The absolute error between estimated and ground truth positions was recorded. As shown in Figure 4-6, the error increases approximately linearly over time, with a slope of about 0.007 $\frac{m}{s}$. Dips in the error curve indicate points where the AUV paused and reversed direction.

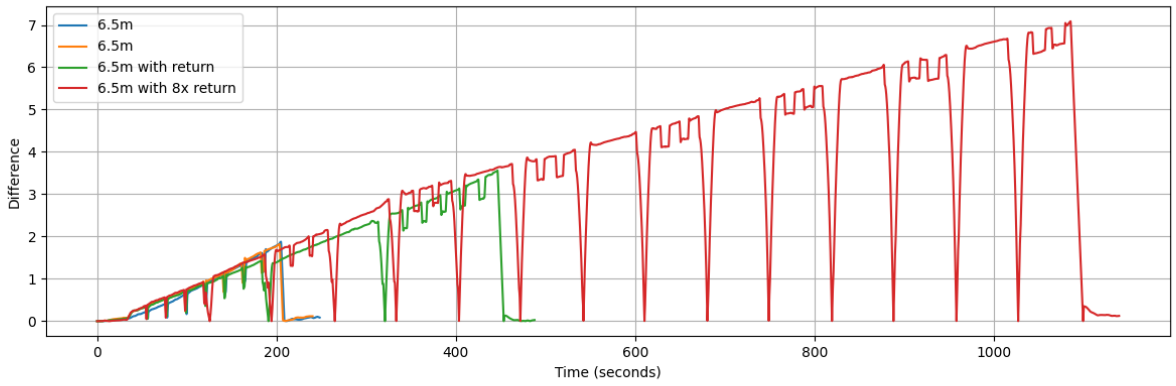


Figure 4-6: Absolute position error during straight-line traversal and return. Green and red represent separate runs. Dips indicate pauses or reversals.

Over a duration of 1200 seconds, the accumulated error reaches approximately 7 meters. This drift suggests that even with map-based guidance, the SLAM system accumulates error—likely due to limited loop closure opportunities and delayed correction mechanisms.
Figures 4-7 and 4-8 show the generated map along the 80-meter trajectory, including height variations.

Measurement Noise

The estimation of position and orientation depends on data from the SLAM algorithm, IMU data, and the barometer, all of which include some noise. This noise is introduced in the ArduSub component, making it not possible to retrieve the specified noise levels, only raw (noise-free) IMU data can be accessed. Weather or not the EKF effectively filters out this noise

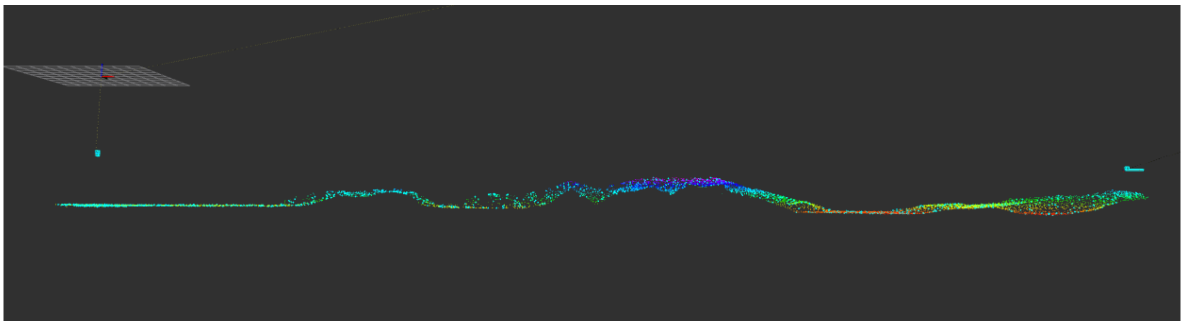


Figure 4-8: Side view of the same 80-meter map. Height differences are color-coded.

is unknown, as evidenced by the discrepancy between the ground truth and the estimated orientation shown in Fig. 4-9.

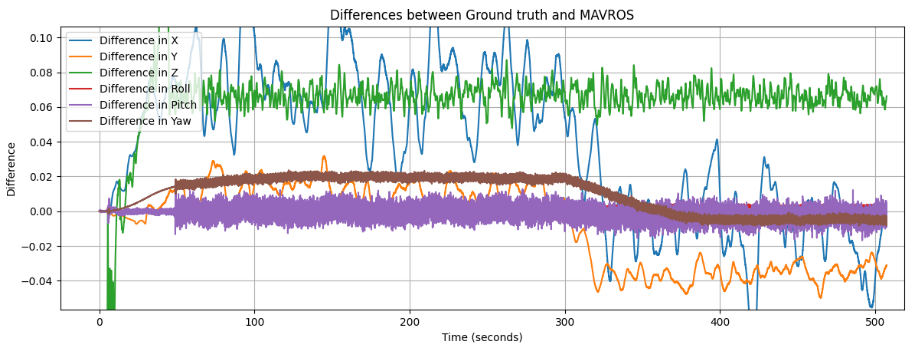


Figure 4-9: Measurement noise when the AUV is moving to -6.5 meters and then remaining stationary

Chapter 5

Modeling and System Identification

The implementation of the SDP-GD algorithm requires a Linear Time-Varying (LTV) system model. However, the dynamics of the Autonomous Underwater Vehicle (AUV) are nonlinear and time-invariant. To bridge this gap and enable the use of LTV-based control strategies, these nonlinear dynamics have to be approximated by a linear model that accurately reflects the system's behavior within specific operating conditions.

The AUV operates under a closed-loop feedback control system. However, in one variation of the SDP-GD algorithm (explained in Chapter 6), the *thruster input* is used directly as the system input. This represents a direct input into the plant, and therefore, an **open-loop system identification** is required. In contrast, for the second input signal variation—where the *reference signal* is used as the input—a **closed-loop identification** approach is necessary due to the presence of feedback in the control structure. This necessitates the need for the modeling of the AUV without the controller.

This chapter discusses the system identification process. It begins with the formulation of a nonlinear model that includes several unknown parameters. To make the model tractable for control design, it is then simplified using a set of assumptions and linearized along a straight-line trajectory. The resulting linear model still contains unknown parameters, which are estimated using system identification techniques based on experimental data. The experimental setup and data processing methods are described as well. Both grey-box and black-box identification approaches are applied and compared at the end.

5-1 Nonlinear Modeling of AUV Dynamics

To develop a control-oriented model of the Autonomous Underwater Vehicle (AUV), it is necessary to first describe its full nonlinear dynamics. The AUV is modeled as a six degrees-of-freedom rigid body, and its motion is described using Newton-Euler equations in Fossen notation [23]. Before deriving the equations of motion, a number of assumptions are made to simplify the physical modeling of the AUV.

Model Assumptions

The following assumptions are adopted for the modeling of the AUV, based on [24]:

Assumption 1: The vehicle is assumed to be a rigid body with six degrees of freedom.
Assumption 2: The AUV is symmetrical along its front-back, left-right, and top-bottom axes, resulting in an even mass distribution.
Assumption 3: The body axes align with its principal axes of inertia. This simplifies the inertia matrix to an identity matrix and decouples the equations of motion.

Assumption 4: The origin of the body-fixed frame is situated at the vehicle's center of mass.
Assumption 5: The ocean current is simulated as a constant, non-rotational flow in the inertial frame. This current affects the vehicle's velocity, while wave effects are neglected.
Assumption 6: The vehicle is assumed to be neutrally buoyant.

Equations of Motion

Using the Newton-Euler formalism, the equations of motion for the AUV are expressed in the standard notation (eq. 5-1) introduced by Fossen [23]. The first part of eq 5-1, $J(\eta)\nu$ is the Jacobian transformation matrix that maps body-frame velocities ν into the inertial frame derivatives of η. The second part of eq 5-1 is the main equation of motion in the body-fixed frame.

$$\begin{aligned} \dot{\eta} &= J(\eta)\nu \\ M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) &= \tau + \tau_w \end{aligned} \tag{5-1}$$

Solving for the acceleration $\dot{\nu}$ yields:

$$\dot{\nu} = M^{-1}\left(-C(\nu)\nu - D(\nu)\nu - g(\eta) + \tau + \tau_w\right) \tag{5-2}$$

where:

- M: mass matrix,
- $C(\nu)$: Coriolis and centripetal matrix,
- $D(\nu)$: damping matrix,
- $g(\eta)$: gravitational and buoyant forces and moments,
- $\nu = [u, v, w, p, q, r]^T$: body-fixed linear and angular velocities (surge, sway, heave, roll, pitch, yaw),
- $\eta = [x, y, z, \phi, \theta, \psi]^T$: positions and Euler angles in the world frame,
- $\tau = [X, Y, Z, K, M, N]^T$: external forces and moments,
- τ_w: unknown external disturbances acting on the system.

Further explanation of each component is provided below can be found in Appendix A-6.

5-2 Linearization

To enable the design of linear controllers and perform system identification, the nonlinear AUV model has to be approximated by a linear system. This is done through linearization, which approximates the nonlinear dynamics around a specific operating point or along a nominal trajectory. Since the AUV is intended to move straight ahead at a constant speed and depth, the x-state will vary over time. Therefore, linearizing around a trajectory is the most suitable approach.

Operating Trajectory and Assumptions

The operating trajectory is defined by the steady-state values of the system's states and inputs. For an AUV moving in a straight line, the steady-state conditions are:

- **Constant forward speed:** The AUV maintains a steady velocity in the x-direction.
- **Negligible lateral and vertical velocities:** The velocities in the y and z directions are approximately zero ($v_0 \approx 0$, $w_0 \approx 0$), assuming minimal disturbances.

- **Constant orientation:** The roll, pitch, and yaw angles remain fixed, so ϕ_0, θ_0, and ψ_0 are constant.
- **Fixed depth:** The AUV operates at a consistent depth, indicated by z_0.
- **Neutral buoyancy:** Assuming that the vehicle is neutrally buoyant, the gravitational force is equal to the buoyancy force: $W = B$ due to the PID controller keeping the z velocity 0.
- **Centered mass and buoyancy:** Given the initial condition is at the center, the center of gravity and buoyancy are set to zero: $x_g = x_b = y_g = z_g = 0$.

Given the initial condition of the AUV $\Big\{x, y, z, \phi, \theta.\psi, u, v, w, p, q, r\Big\} = \Big\{0, 0, -5, 0, 0, 0, 0, 0, 0, 0, 0, 0\Big\}$ and assuming that the system navigates with a constant velocity in the x-direction, the following operating trajectory is defined:

$$\Big\{x, y, z, \phi, \theta.\psi, u, v, w, p, q, r\Big\} = \Big\{x, 0, -5, 0, 0, 0, 1, 0, 0, 0, 0, 0\Big\} \tag{5-3}$$

Linearized System Model Using the Jacobian

The linearization is based on the nonlinear dynamics of the system, expressed as:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \dot{\nu}(\mathbf{x}, \mathbf{u})$$

with the state vector $\mathbf{x}$:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \\ &= \begin{bmatrix} x\ y\ z\ \phi\ \theta\ \psi\ u\ v\ w\ p\ q\ r \end{bmatrix}^T \end{aligned} \tag{5-4}$$

and input vector $\mathbf{u}$:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}, \tag{5-5}$$

The linearized system around the operating point $(\mathbf{x}_0, \mathbf{u}_0)$ becomes:

$$\delta\dot{\mathbf{x}} = A\delta\mathbf{x} + B\delta\mathbf{u} \tag{5-6}$$

where:

- $\mathbf{x}[k] \in \mathbb{R}^{12}$ is the state vector,
- $\mathbf{u}[k] \in \mathbb{R}^{6}$ is the input vector,
- $A \in \mathbb{R}^{12\times12}$ is the state matrix,
- $B \in \mathbb{R}^{12\times6}$ is the input matrix.

The matrices A and B are obtained from the Jacobian of the nonlinear dynamics with respect to the state and input:

$$A = J_x = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0}, \quad B = J_u = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0} \tag{5-7}$$

Here, the full Jacobian matrix J is composed of first-order partial derivatives with respect to both state and input:

$$J = \begin{bmatrix} J_x & J_u \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} & \frac{\partial f_2}{\partial u_1} & \cdots & \frac{\partial f_2}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} & \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}$$

Using the assumptions and trajectory, the Jacobian matrices take the following forms:

$$A = \begin{bmatrix} 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&-1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&\frac{X_u+2X_{uu}}{m}&0&0&0&0&0 \\ 0&0&0&0&0&0&0&\frac{Y_v}{m}&0&0&0&-1 \\ 0&0&0&0&0&0&0&0&\frac{Z_w}{m}&0&1&0 \\ 0&0&0&0&0&0&0&0&0&\frac{K_p}{I_{xx}}&0&0 \\ 0&0&0&0&0&0&0&0&0&0&\frac{M_q}{I_{yy}}&0 \\ 0&0&0&0&0&0&0&0&0&0&0&\frac{N_r}{I_{zz}} \end{bmatrix}$$

(a) State matrix A (Jacobian of the system with respect to the states)

$$B = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ \frac{1}{m}&0&0&0&0&0 \\ 0&\frac{1}{m}&0&0&0&0 \\ 0&0&\frac{1}{m}&0&0&0 \\ 0&0&0&\frac{1}{I_{xx}}&0&0 \\ 0&0&0&0&\frac{1}{I_{yy}}&0 \\ 0&0&0&0&0&\frac{1}{I_{zz}} \end{bmatrix}$$

(b) Input matrix B (Jacobian of the system with respect to the inputs)

Figure 5-1: Linearized system matrices used for state-space modeling.

From the linearized transition matrix A, the following relationships can be extrapolated:

- The x-position depends only on the forward velocity u.
- The y-position depends on the lateral velocity v and the yaw angle ψ.
- The z-position depends on the vertical velocity w and the pitch angle θ.
- The orientation angles ϕ, θ, ψ are dependent on their respective angular velocities p, q, r.
- The velocity u depends only on itself and its corresponding input reference.
- The velocity v depends only on the yaw rate r and its input reference.
- The velocity w depends on the pitch angle θ and its input reference.
- The angular velocities p, q, r are each dependent on themselves and their respective input references.

Given that the AUV does not rotate during straight-line motion (i.e., $\delta p = \delta q = \delta r = 0$) and that the initial angles are zero and remain approximately constant throughout the simulation, the angles can be removed from the state vector. Similarly, velocities can be substituted into position derivatives, simplifying the state-space representation to:

$$x = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Inputs: Later in this report, the implementation of the SDP-GD algorithm will be discussed in detail, including two variations based on the type of input signals used: one with control input and one with control input. The control input corresponds to the raw thruster signals controlling velocity in the x, y, and z directions. In contrast, the control Input consists of the reference velocity commands in these directions.

This distinction leads to the following input signal structure:

$$u = \begin{bmatrix} 0 & 0 & 0 & u_x & u_y & u_z \end{bmatrix}^T \tag{5-8}$$

The first three elements are set to zero because the input signals only affect the translational velocities. The position of the AUV is then determined by integrating these velocity inputs over time

Discretization: The position dynamics resemble simple integrators, while velocity dynamics depend on themselves and their respective inputs. For system identification, the model is implemented in MATLAB using the `idgrey` function, which allows for both continuous- and discrete-time system definitions. For discrete models, a sampling time must be provided.

Time-Varying: Although the model structure is fixed, the system is implemented in a time-varying form since the linearization is performed along a trajectory. This allows the transition matrices to remain constant for all time steps while still capturing time-varying behavior through state updates.

Parameterization and Model Structure: In the matrices A and B, each unknown element could be parameterized individually. However, expressing each element as a separate parameter (e.g., $\frac{\theta_1}{\theta_2} = \frac{Y_v}{m}$) unnecessarily complicates the model structure. Instead, the complete element is treated as a single parameter, e.g., $\frac{Y_v}{m} = \theta_1$.

In theory, velocities are independent due to earlier assumptions. In reality, however, coupling exists. Therefore, in the grey-box model, dependencies between velocities are included in the parameterized form.

Grey-Box Model

For the grey-box parameterization, certain assumptions are made that lead to a 6×6 system with a simplified structure, resulting in a total of 18 identifiable parameters across the A and B matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \delta t \\ 0 & 0 & 0 & \theta_1 & \theta_2 & \theta_3 \\ 0 & 0 & 0 & \theta_4 & \theta_5 & \theta_6 \\ 0 & 0 & 0 & \theta_7 & \theta_8 & \theta_9 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \theta_{10} & \theta_{11} & \theta_{12} \\ 0 & 0 & 0 & \theta_{13} & \theta_{14} & \theta_{15} \\ 0 & 0 & 0 & \theta_{16} & \theta_{17} & \theta_{18} \end{bmatrix}$$

The intuition behind this parameterization is that the position states are influenced by the velocity states, but not vice versa—meaning velocity affects position, but position does not directly influence velocity in the system dynamics.

Since the input vector u contains zeros in the first three elements (representing no control input directly affecting position), the left portion of the B matrix can be set to zero without loss of generality.

Black-Box Model

As a fallback or validation strategy, a fully parameterized black-box model is implemented with 72 parameters. This model will also be used for the closed loop identification. This model allows each element of the A and B matrices to vary independently, with no structural assumptions:

$$A = \begin{bmatrix} \theta_1 & \theta_2 & \dots & \theta_6 \\ \vdots & \ddots & \dots & \vdots \\ \theta_{31} & \dots & \dots & \theta_{36} \end{bmatrix}, \quad B = \begin{bmatrix} \theta_{37} & \theta_{38} & \dots & \theta_{42} \\ \vdots & \ddots & \dots & \vdots \\ \theta_{67} & \dots & \dots & \theta_{72} \end{bmatrix}.$$

The grey-box and black-box models are compared against each other and chosen to apply for the SDP-GD algorithm.

5-3 Experimental Setup of Identification Process

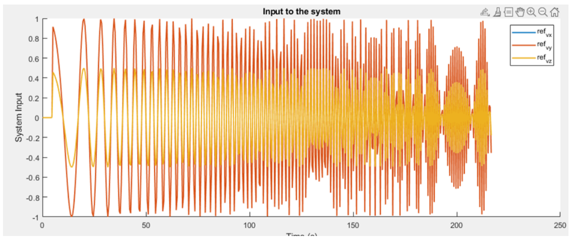
In the identification experiments, the AUV is initially positioned at $(x, y, z, \phi, \theta, \psi) = (0, 0, -5, 0, 0, 0)$. An input signal is applied to the AUV, and the resulting output data is collected. For validation, a step input is introduced to simulate forward motion, which matches the conditions used later in the algorithm comparison. The system identification is performed using MATLAB. The following sections describe the input and output signals used in the process.

Input Signals for System Identification

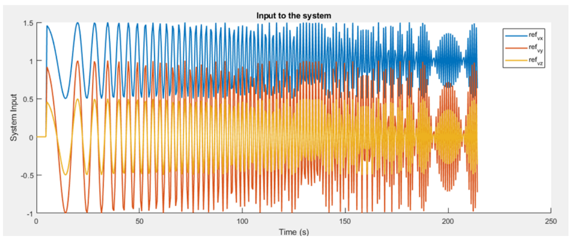
To sufficiently excite the system and identify its full dynamics, two sets of chirp signals are used as input. In both cases, the chirp signals are applied simultaneously to the reference velocities in the x-, y-, and z-directions ($v_{x_\text{ref}}, vy_\text{ref}, vz_\text{ref}$) as input signal, but with different amplitudes and mean values to assess identification performance under different conditions.

In the first scenario (Figure 5-2a), a chirp signal with amplitude 1 and zero mean is applied to the x-velocity. The y-velocity is also excited with an amplitude of 1 and zero mean, while the z-velocity uses a chirp signal with amplitude 0.5 and zero mean. Since the signals have zero mean, the AUV moves back and forth around its initial position in all three directions.

In the second scenario (Figure 5-2b), the x-velocity is excited using a chirp signal with an amplitude of 0.5 and a mean value of 1, while the y- and z-velocities remain centered around zero mean with amplitudes of 1 and 0.5, respectively. In this case, the AUV moves forward with oscillating velocity, while also oscillating vertically, resulting in a trajectory that combines forward motion with small vertical displacements around 0.5 m/s.



(a) Scenario 1: vx_ref and vy_ref with amplitude 1 and zero mean; vz_ref with amplitude 0.5 and zero mean.



(b) Scenario 2: vx_ref with amplitude 0.5 and mean 1; vy_ref with amplitude 1 and zero mean; vz_ref with amplitude 0.5 and zero mean.

Figure 5-2: Chirp input signal configurations used for system identification in two different scenarios.

Table 5-1: RMSE for Identification Using Chirp(0,1) and Chirp(1,0.5)

	Chirp(0,1)		Chirp(1,0.5)	
Output	Grey-Box RMSE	Black-Box RMSE	Grey-Box RMSE	Black-Box RMSE
y_1	0.54	1.18	0.44	0.92
y_2	0.05	0.08	0.01	0.01
y_3	0.47	1.08	0.25	0.25
y_4	0.36	0.35	0.24	0.24
y_5	0.00	0.00	0.00	0.00
y_6	0.21	0.21	0.21	0.22

Output Signals

The system outputs consist of velocity measurements in the x, y, and z directions, obtained using the onboard IMU and DVL. The corresponding positions in each axis are calculated by integrating these velocity signals over time. To identify each state, the output matrix is set to identity leading to using the measurements:

$$(y_1, y_2, y_3, y_4, y_5, y_6) = (x, y, z, v_x, v_y, v_z)$$

5-4 Results

This section compares the Grey-Box and Black-Box modeling approaches by evaluating their performance in both the identification and validation phases. The comparison is based on Root Mean Square Error (RMSE) values.

Identification Results

The System Identification Toolbox provides a model fit percentage as a performance measure. Table B-3 in the appendix presents these results for both models, using two different chirp input scenarios.

Although the model fit percentage is a commonly used metric, it can be misleading in this context. For example, a fit of -13.86% may appear poor, but it corresponds to an RMSE of 0.32, which is within an acceptable range. Therefore, RMSE is also used as a more direct and interpretable measure of error. The corresponding RMSE values are shown in the table below:

Validation Results

For validation, a step input is applied to the x-velocity, inducing a dynamic response in the AUV and allowing for further evaluation of model accuracy. The tables below present the model fit percentages and RMSE values for each system state.

Discussion

From the initial identification results, the Grey-Box model demonstrates better performance than the Black-Box model in both the identification and validation phases. However, the improvement during identification is marginal, with both models showing similar performance

Table 5-2: RMSE for Validation Using Chirp(0,1) and Chirp(1,0.5)

	Chirp(0,1)		Chirp(1,0.5)	
Output	Grey-Box RMSE	Black-Box RMSE	Grey-Box RMSE	Black-Box RMSE
y_1	17.22	25.58	0.31	0.57
y_2	0.01	0.82	0.05	0.03
y_3	1.84	12.21	0.17	0.17
y_4	1.64	5.13	0.13	0.13
y_5	0.02	0.03	0.00	0.00
y_6	0.70	4.02	0.02	0.05

for both input scenarios — Chirp(0, 1) and Chirp(1, 0.5). This was somewhat unexpected, as the Black-Box model was initially expected to perform comparably during identification.

The performance gap becomes more apparent during validation. For the Chirp(0, 1) scenario, the Grey-Box model clearly outperforms the Black-Box model. For Chirp(1, 0.5), the Grey-Box model still performs slightly better, although the difference is less pronounced.

Based on these findings, the model identified using the Chirp(1, 0.5) input and the Grey-Box approach will be used for the subsequent algorithm comparison.

It should be noted that a maximum of 300 iterations was used for the parameter estimation process. The results may vary depending on the initial parameter values, which could lead to different identification outcomes.

5-5 BlueROV2 System identification

This section describes the implementation of a closed-loop state-space model using SLAM data from Orb-SLAM2, with the goal of identifying system matrices and simulating the behavior of the BlueROV2. The AUV will have 2 main trajectories, i.e. going down to a certain depth and going forward for a certain amount of time at a fixed depth (6.5 meters below sealevel).

Closed-Loop Identification

From the simulation results, it can be observed that the trajectory has a linear behavior. This is mainly due to the presence of a controller, which stabilizes the system and suppresses nonlinear dynamics. For the purpose of system identification, a simple step input was applied. However, it should be noted that this type of input does not excite the system as to an order as a chirp signal would, potentially limiting dynamics and the accuracy of the identified model.

Reference Values and Initial Conditions

Both trajectories are handled as if they were 2 different datasets for the identification. The data is cut at $t_{\text{cutoff}} = 25$ seconds. This time is chosen based on the time the AUV reaches the depth of 6.5 meters. For the identification of the 2 different trajectories, 2 different initial conditions are used for the solver and 2 different reference signals, i.e.

- Before t_{cutoff}: reference $= (0, 0, -6.5)$, $\mathbf{x}_0 = [0, 0, 0]$
- After t_{cutoff}: reference $= (100, 0, -6.5)$, $\mathbf{x}_0 = [0, 0, -6.5]$

Input Signal

The system operates in closed loop, where the input is the reference position vector $u_{\text{ref}}[k]$, and the output $y[k]$ is the measured position from SLAM:

$$u_{\text{ref}}[k] = \begin{bmatrix} x_{\text{ref}}[k] \\ y_{\text{ref}}[k] \\ z_{\text{ref}}[k] \end{bmatrix}, \quad y[k] = \begin{bmatrix} x[k] \\ y[k] \\ z[k] \end{bmatrix}$$

To define the system input $u[k]$ for identification, the reference signal is subtracted from the measured trajectory:

$$u[k] = u_{\text{ref}}[k] - y[k] \tag{5-9}$$

Parameterization

To to a linear behavior a simple linear regression is used. The system is modeled using a discrete-time state-space representation, defined by:

$$\dot{x} = Ax(\theta) + Bu \tag{5-10}$$

$$A[k] = \begin{cases} \begin{bmatrix} \theta_1 & 0 & 0 \\ 0 & \theta_2 & 0 \\ 0 & 0 & \theta_3 \end{bmatrix}, & \text{for } k \in [0,5] \\ \begin{bmatrix} \theta_7 & 0 & 0 \\ 0 & \theta_8 & 0 \\ 0 & 0 & \theta_9 \end{bmatrix}, & \text{for } k > 5 \end{cases} \tag{5-11}$$

$$B[k] = \begin{cases} \begin{bmatrix} \theta_4 & 0 & 0 \\ 0 & \theta_5 & 0 \\ 0 & 0 & \theta_6 \end{bmatrix}, & \text{for } k \in [0,5] \\ \begin{bmatrix} \theta_{10} & 0 & 0 \\ 0 & \theta_{11} & 0 \\ 0 & 0 & \theta_{12} \end{bmatrix}, & \text{for } k > 5 \end{cases} \tag{5-12}$$

where:

- $x[k] \in \mathbb{R}^3$ is the state vector, which includes the x, y, z position of the robot.
- $u[k] \in \mathbb{R}^3$ is the input vector,
- $A \in \mathbb{R}^{3\times3}$ is the state matrix, modeled as a diagonal matrix with parameters θ_i,
- $B \in \mathbb{R}^{3\times3}$ is the input matrix, also diagonal, with parameters θ_j.

Cost Function for Optimization

A cost function is defined to evaluate the difference between the simulated trajectory and the measured SLAM data. The error is computed as:

$$J(\theta) = \|x_{\text{sim}}[k] - y_{\text{measured}}[k]\|_2 \tag{5-13}$$

where $y_{\text{measured}}[k]$ is the observed SLAM trajectory and $\|\cdot\|_2$ represents the Euclidean norm. The goal is to minimize $J(\theta)$ by finding optimal parameters θ for matrices A and B.

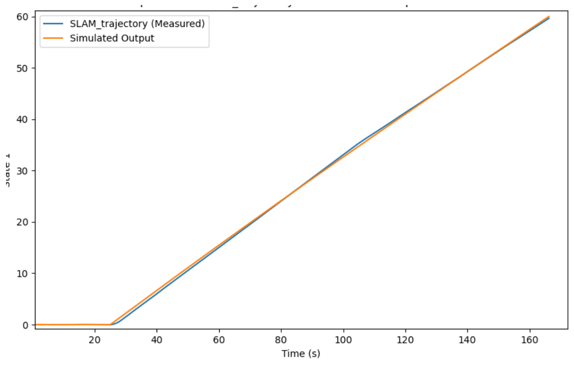
The system dynamics are simulated using the solver `solve_ivp`. Given initial conditions x_0, the integration spans the time interval $t \in [0, T]$, resulting in the simulated trajectory $x_{\text{sim}}[k]$.

Validation

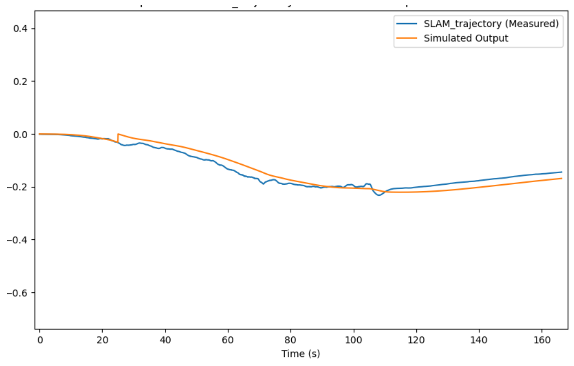
The output $y_{\text{sim}}[k]$ is computed as:

$$y_{\text{sim}}[k] = Cx_{\text{sim}}[k] \tag{5-14}$$

where $C = I$ (identity matrix), resulting in $y_{\text{sim}}[k] = x_{\text{sim}}[k]$. This is then compared against the SLAM trajectory $y_{\text{measured}}[k]$ through visual plots.



(a) Simulated X Axis (orange) vs actual X Axis (blue)



(b) Simulated Y Axis (orange) vs actual Y Axis (blue)

Figure 5-3: Comparison of simulated and actual signals for X and Y axes

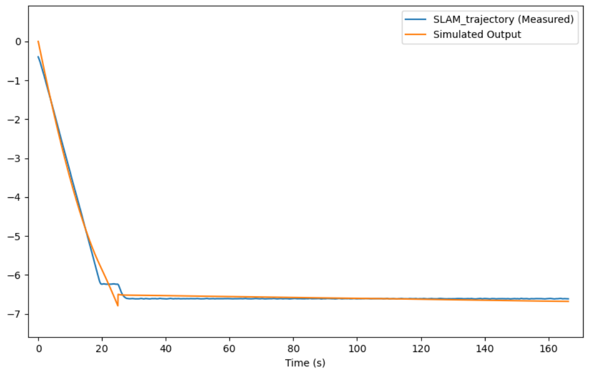


Figure 5-4: Simulated Z Axe (orange) vs actual Z Axe (Blue)

Table 5-3: Mean Error for Each State

State	**Mean Error**
x	2.258
y	-0.006
z	0.533

Chapter 6

SDP-GD Implementation

This chapter outlines the design decisions made for implementing the MAP estimation algorithm using the SDP-GD approach. The implementation includes the setup of noise parameters and initial conditions. The SDP-GD algorithm is extended to support three different output maps, which are individually computed and subsequently merged.

Additionally, the implementation of the Sonar-Based SLAM algorithm is described, including the variations introduced for the comparison in the next section. A similar structure is then applied for the implementation and evaluation of the Visual-Based SLAM system.

6-0-1 Pearson correlation and Coefficient of determination

A measure to evaluate the goodness of fit between 2 signals (i.e. depth reconstruction) the **Pearson correlation** is used, which is described as:

$$r = \rho_{X,Y} = \frac{\mathbb{E}\left[(X - \mu_X)(Y - \mu_Y)\right]}{\sigma_X \sigma_Y} \tag{6-1}$$

$$\begin{aligned}
\rho_{X,Y}&: \text{Pearson correlation coefficient between } X \text{ and } Y \\
\mu_X &= \mathbb{E}[X] : \text{Mean of } X \\
\mu_Y &= \mathbb{E}[Y] : \text{Mean of } Y \\
\sigma_X &= \text{Standard deviation of } X \\
\sigma_Y &= \text{Standard deviation of } Y
\end{aligned}$$

Pearson correlation can have a value between [-1,1], with

- $r = 1$: perfect positive linear correlation
- $r = -1$: perfect negative linear correlation
- $r = 0$: no linear correlation

Squaring the Pearson correlation, results in a measure with value between 0 and 1, which indicates for $R^2 = 1$ a perfect linear correlation and value of $R^2 = 0$ indicates no linear correlation. The squared Pearson correlation is known as **Coefficient of determination** [25]

6-1 Downsampling and Aliasing

Sensor data in the simulator is collected at varying sampling rates (see Appendix B-1). To ensure temporal consistency, this data must be aligned to a common sampling rate—typically requiring downsampling. A known risk of downsampling is **aliasing**, where high-frequency components are incorrectly represented as lower frequencies.
While upsampling via interpolation is possible, it introduces synthetic data that may distort the signal. For this reason, it is generally preferable to work with sparse but accurate measurements.

A FFT analyses is performed for the different seabed complexity. Which the highest seabed complexity yields the highest frequency (bandwidth). The FFT for seabed complexity 12 is seen in the following figure:

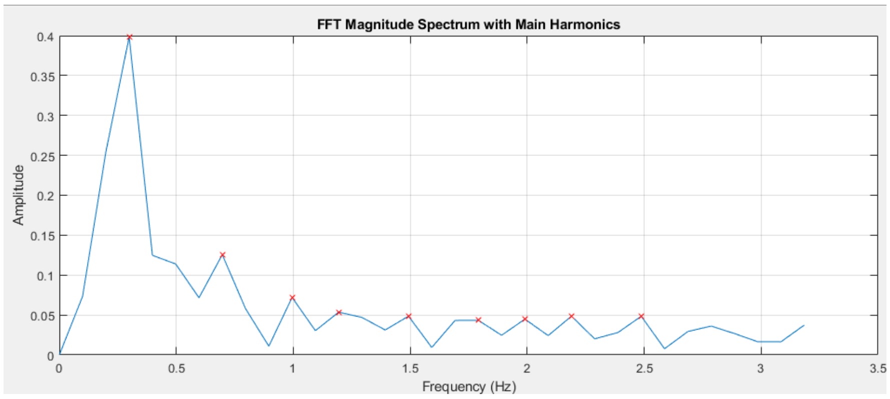


Figure 6-1: FFT Analyses of the seabed with complexity level 12

From the FFT analyses the highest frequency component is equal to $f_s = 2.5Hz$, therefore a sampling frequency of 6Hz is used, which is $6Hz > 2f_s$.

6-2 General Implementation Details

Data Requirements

This section explains the data required for applying the SDP-GD algorithm. The following lifted-form data inputs are necessary:

- Σ_v: The measurement noise covariance matrix.
- Σ_x: The process noise covariance matrix. This is identified during the system identification phase.
- $X_{\textbf{nominal}}$: The nominal state prediction vector, derived from the lifted process model:

$$X_{\text{nominal}} = A_{\text{lifted}} \cdot u_{\text{lifted}}. \tag{6-2}$$

  In this case, the input u_{lifted} is the reference signal for the desired velocity.
- $y_{\textbf{measured}}$: The measured output vector.
- $C_{\textbf{nominal}}$: A vectorized form of the nominal observation matrix C.
- Σ_C: The covariance matrix of the output map C. This parameter is based on the unknown parameters noise covariance. It is set to 0.01 as a low arbitrary value to reflect a small expected estimation error. Although this value can be set to zero, in practice it is mostly nonzero.

- **desiredSnr**: The desired signal-to-noise ratio used for weighting or scaling the estimation.
- **matrixScale**: A scaling factor applied to matrices during estimation.

Noise Parameters and Covariance Tuning

The initial parameter covariance Σ_{θ_0}, representing the uncertainty in the initial seabed model, is set to 4. This is based on the assumption that the seabed can vary by approximately 4 meters. The process noise Σ_v is set equal to the state process noise Σ_x.

The output map covariance matrix, Σ_C, is set to an arbitrary low value of 0.01, representing a low expected estimation error. This value reflects the assumption that the measured signal is relatively close to the ground truth. In practical underwater scenarios, obtaining accurate measurements of the difference between estimated and true positions is extremely challenging without expensive high-precision systems like USBL (Ultra-Short Baseline) or LBL (Long Baseline) acoustic positioning.

Initial Conditions and Signal Alignment

The system's initial condition is set to zero for all states, assuming that the AUV's spawn location in the simulator represents the initial state. Since the sonar measures the distance to the seabed, the entire measured signal is offset by subtracting the initial depth estimation. This ensures that the depth data starts from zero.

This alignment step is critical because the depth is estimated by multiplying the state with the output map C. If the initial state is zero and the signal is not offset, the depth estimate will also be zero, leading to false detection or misinterpretation of the seabed structure.

Combining Multiple Output Maps

The output map $C(\theta)$ can be constructed with multiple rows to estimate multiple outputs, such as x, y, and d (depth). However, directly including all outputs in the same optimization problem increases computation time and prevents multithreading.

To address this, the SDP-GD approach computes each row of the $C(\theta)$ matrix independently. Afterward, the results are combined by vertically stacking each dimension-specific result into a single matrix:

$$C_i = \begin{bmatrix} C_i(1) \\ \vdots \\ C_i(n_T) \end{bmatrix} \quad \text{for } i \in \{x, y, h\},$$

$$C = \begin{bmatrix} C_x(1) \\ C_y(1) \\ C_h(1) \\ \vdots \\ C_x(n_T) \\ C_y(n_T) \\ C_h(n_T) \end{bmatrix} \tag{6-3}$$

This modular approach allows for parallel computation and scalability while preserving the estimation quality for each output.

6-3 Sonar-Based SLAM Implementation

For the application of the SDP-GD algorithm, four variations are explored, each using different input signals and measurement types. In the final variation, a modification is introduced to further improve the computation time of the algorithm.

Variation 1: C_{var_1} control Input

In this variation, the controller's reference signal, a step input with an amplitude of 1, is used.

Motivation: This choice ensures consistency across all experiments, as the input signal remains constant. However, the output map produces a repeatable and reliable response. Such consistency may be beneficial in applications that require consistent system behavior.

The state vector is defined as:

$$x = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T$$

where x, y, and z represent the position states, which do not depend on the reference signal.

These position states can be initialized arbitrarily, since the corresponding entries in the B matrix are zero. This means they are not directly influenced by the input signal, which is:

$$u = \begin{bmatrix} 0 & 0 & 0 & u_{v_{x_\text{ref}}} & u_{v_{y_\text{ref}}} & u_{v_{z_\text{ref}}} \end{bmatrix}^T$$

The measurement in this variation is the position of the AUV, obtained by integrating the velocity estimated from the DVL and IMU sensors and the depth (h) to the seabed is measured using the sonar data:

$$y_\text{meas} = \begin{bmatrix} x & y & h \end{bmatrix}^T$$

This integration process corresponds to dead reckoning. From experiments, the integrated velocity closely matches the ground truth position, assuming minimal sensor noise. However, in real-world scenarios, sensor noise can lead to significant drift over time.

Variation 2: C_{var_2} Control input

In this variation, the state and measurement signals remain the same as in Variation 1. The input is changed to the thruster control signals, which directly influence the AUV's measurements.

Motivation: This choice also include the necessary adjustment due to environmental disturbances. The Destik SAGA ROV is equipped with three thrusters, resulting in three input signals for the SDP-GD algorithm. As in the identification procedure, the input signal, corresponding to the x, y, and z positions, is set to zero.

$$u = \begin{bmatrix} 0 & 0 & 0 & u_\text{thruster 1} & u_\text{thruster 2} & u_\text{thruster 3} \end{bmatrix}^T$$

A third variation, suggested by S. Vakili, was also implemented. However, this modification led to only minimal improvement in seabed reconstruction performance. Details of this variation can be found in Appendix C.

Experiments were performed to Given the minimal and inconclusive improvements in depth estimation, combined with the additional computational cost of generating a second output map, this variation is not used in the final comparison of the SLAM algorithms.

6-4 Visual SLAM

Limitation

A proposed method for applying visual SLAM, inspired by the Sonar-Based SLAM approach, involves using the onboard camera to detect keypoints which can then be utilized to construct a point cloud. However, a major challenge arises from the nature of the ORB-SLAM2 algorithm, which outputs a vector of descriptors without explicitly revealing the spatial positions of individual keypoints. To overcome this, the SDP-GD will be used as a smoother, using the output of ORB-SLAM2 to improve seabed reconstruction.

Experiments

Measurement are taken to apply to the SDP-GD to measure its performance for the different variances. The experiment involves four predefined trajectories, each executed twice for three different environments (maps), resulting in a total of 24 runs. Each trajectory directs the vehicle toward one of the following target destinations:

- Trajectory 1: $(x, y, z)_{\text{Destination}} = (50, 50)$
- Trajectory 2: $(x, y, z)_{\text{Destination}} = (50, -50)$
- Trajectory 3: $(x, y, z)_{\text{Destination}} = (-50, 50)$
- Trajectory 4: $(x, y, z)_{\text{Destination}} = (-50, -50)$

Variation 1

This refinement process does not fall under the definition of a SLAM algorithm, as it neither performs simultaneous localisation nor real-time mapping. Instead, it enhances an existing map, treating the output of ORB-SLAM2 as a preliminary estimate.

Measurement: The measurement used for the SDP-GD becomes:

$$y_{\text{meas}} = d_{\text{OrbSlam2}} \tag{6-4}$$

Input: The system is in a feedback loop, where the input into the system is a reference signal for the desired position of the AUV for the x, y and z position .

$$u = \begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \\ z_{\text{ref}} \end{bmatrix} \tag{6-5}$$

It is important to note that ORBSLAM2 does not rely solely on an input trajectory. Rather, it depends on a combination of the input command, visual information from the camera, and the ORB-SLAM2 algorithm to navigate. Therefore, in the evaluation, the seabed reconstruction generated by ORB-SLAM2 will be compared against the reconstruction obtained using the SDP-GD algorithm, which, for this run, uses only the input trajectory of the current test to generate a reconstructed map.

Contribution: The advantage with the SDP-GD method is that the AUV can execute its trajectory without requiring camera input or running ORB-SLAM2 in real-time. This can lead to substantial energy savings, making the system more efficient for long-duration missions.

The output map for this variation has the following form:

$$C_{\text{var }1}(\theta) = f(y,u) = f(d_{\text{OrbSlam2}}, \begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \\ z_{\text{ref}} \end{bmatrix}) \tag{6-6}$$

In Figure 6-2, this variation of the SDP-GD algorithm is applied, and the resulting seabed reconstruction is compared with that of ORB-SLAM2. It can be observed that the SDP-GD reconstruction closely follows the ORB-SLAM2 signal, due to using the output of the ORB-SLAM2 as measurement for the SDP-GD.

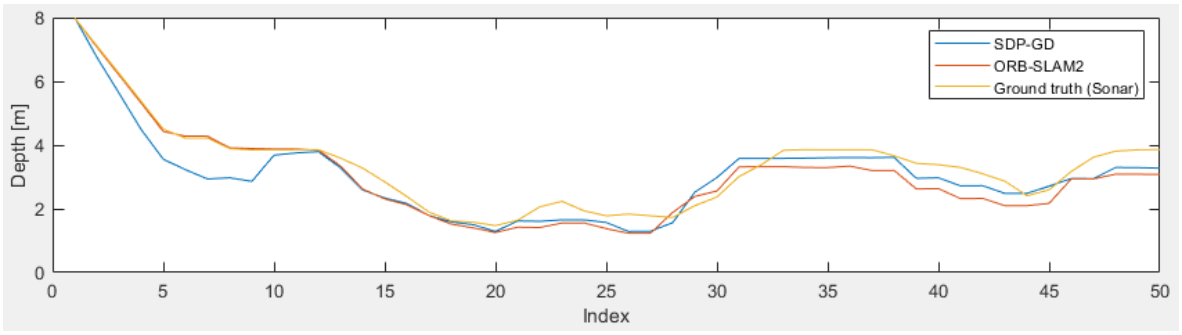


Figure 6-2: Seabed reconstruction SDP-GD outmap variation 1 (blue) and ORB-SLAM2 (red) compared to the ground truth (yellow).

It can also be observed in Figure 6-2 that at the beginning of the sequence ($k = 2 \ldots 10$), there is a noticeable dip in the SDP-GD reconstruction. This dip may be attributed to poor identification of that segment of the system. One possible explanation is that the model began to move prior to the specified time segment in the time-varying model, which was set to begin at $k = 5$.
By applying the full set of measurements to this variation, a more comprehensive analysis can be conducted. Table 6-3 presents the Mean Squared Error (MSE) and Pearson Correlation values for the SDP-GD, across different map complexities and trajectories.
Table 6-1 will be used as a benchmark for Variation 2.

Variation 2

This variation similiar to the first variation, however the measurement here is different. Here the measurement is instead of the depth estimate the measurement taken by a sonar sensor.

Therefore this variation has the following output map:

$$C_{\text{var }2}(\theta) = f(y,u) = f(d_{\text{Sonar}}, \begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \\ z_{\text{ref}} \end{bmatrix}) \tag{6-7}$$

An example of the seabed reconstruction using the output map generated from Map 0 and Trajectory 1 is shown in Figure 6-3. In this figure, the reconstructed seabed is visualized

Table 6-1: MSE and Pearson Correlation for SDP-GD using Variation 1 for visual SLAM, across different map complexities and trajectories

Complexity	**Trajectory**			
	1	**2**	**3**	**4**
MSE SDP-GD				
0	0.19163	0.35191	0.75149	0.15184
1	0.9895	3.1358	24.302	0.68021
3	0.31791	0.1061	9.2048	2.4455
Pearson Correlation SDP-GD				
0	0.95999	0.76664	0.81007	0.92083
1	0.95394	0.59015	0.9568	0.90888
3	0.85252	0.8287	0.64933	0.5574

for three cases: the SDP-GD algorithm (blue), ORB-SLAM2 (red), and the ground truth (yellow).

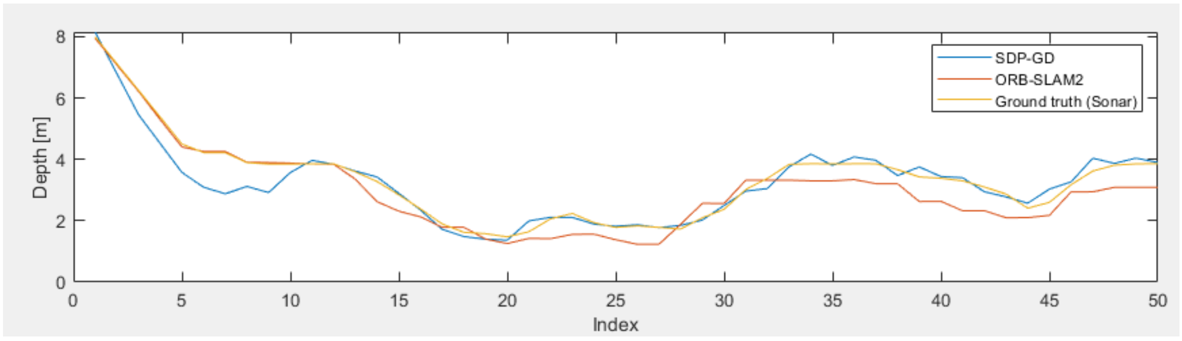


Figure 6-3: Seabed reconstruction comparison using data from Map 0 and Trajectory 1. The plot shows the output of the SDP-GD algorithm (blue), ORB-SLAM2 (red), and the ground truth (yellow).

This seabed reconstruction is similiar to fig 6-2, which shows mayor improvement, especially that the SDP-GD follows the ground truth data.

Apply the set of measurement on this variation has the following result:

Compared to variation 1 (see Table 6-1), variation 2 performs better, showing a Pearson correlation coefficient and a lower mean squared error (MSE). Therefore, this variation will be used for the comparison of the SDP-GD and the Visual SLAM algorithm in combination with the third variation.

Variation 3

In this variation, the SDP-GD algorithm is modified to increase the computation of the output map by optimizing it in smaller batches. In both the first and second variations, the time horizon is set to $n_T = 50$, and, as described in Equation 7-1, the computational complexity scales with n_T^3. To address this, the modified approach reduces the time horizon to $n_T = 1$ by using a batch size of 1. This configuration is chosen to represent the minimal possible batch

Table 6-2: MSE and Pearson Correlation for SDP-GD using Variation 2 for visual SLAM, across different map complexities and trajectories

Complexity	**Trajectory**			
	1	**2**	**3**	**4**
	MSE SDP			
0	0.29859	0.58760	0.63284	0.23941
1	0.86671	2.57800	24.9700	0.98950
3	0.97573	0.58253	11.3350	1.98260
	Pearson Correlation SDP			
0	0.96706	0.88800	0.89767	0.91736
1	0.96421	0.91905	0.40072	0.99716
3	0.90845	0.86094	0.87922	0.91964

Table 6-3: Comparison of Computation Time, Optimal Value, and Iteration Count for SDP-GD and Batched SDP-GD (Batch Size = 1) Across All Trajectories and Maps

Case	**Comp. Time**	**Opt. Value**	**Iter.**	**Batches Time**	**Batches Opt. Value**	**Batches Iter.**
Map 1, traj 1	19.492	-78.841	358	22.471	-78.841	273
Map 2, traj 1	26.486	-82.865	446	22.579	-82.865	276.5
Map 3, traj 1	24.180	-83.243	395.5	23.054	-83.243	267.5
Map 1, traj 2	19.782	-77.457	356.5	22.316	-77.457	280
Map 2, traj 2	24.095	-89.106	384	22.644	-89.106	265
Map 3, traj 2	19.728	-82.965	418	17.092	-82.965	293.5
Map 1, traj 3	18.862	-78.009	333	19.897	-78.009	253.5
Map 2, traj 3	23.030	-72.600	357.5	20.185	-72.600	206
Map 3, traj 3	18.154	-78.738	364.5	16.762	-78.738	199.5
Map 1, traj 4	19.111	-76.561	234.5	25.852	-76.561	305.5
Map 2, traj 4	28.802	-81.493	510	24.420	-81.493	310.5
Map 3, traj 4	19.718	-87.357	416	18.007	-87.357	312.5

size, allowing for the assessment of its impact on computational performance. Additionally, the maximum number of iterations during the optimization step is capped at 100 to further limit computation time.

Variation 2 vs 3

In this section, the SDP-GD algorithm is applied, and its computational efficiency is evaluated.

The same set of trajectories is used across all three maps, meaning the trajectories are identical in each environment. For every trajectory and map combination, an output map is generated using both variation 2 and variation 3 of the SDP-GD algorithm (where variation 3 utilizes batch processing with a batch size of 1).
To evaluate computational performance, the following metrics are recorded for each run: Computation time (in seconds), Optimal value of the cost function and Number of iterations to convergence.
Afterward, averages are computed across the runs for comparison. Table 6-3 presents the results for each case.

It can be observed that both algorithm variations consistently converge to the same optimal cost value. However, differences in computational time and the number of iterations are notable between the two approaches. The output maps were calculated in a parallel loop, which might effect the computational time. The difference between the computational time using batches are generally lower than of the SDP-GD, but still marginally ranging from -15% slower to 31% faster. Due to the optimal Value Cost resulting in the same number and the computational time being generally lower, the 3rd variation will be used for the comparison of the SDP-GD to visual SLAM.

Chapter 7

Comparative Analysis and Evaluation

When comparing two algorithms for seabed reconstruction, it is essential to evaluate them against a comprehensive set of criteria. These criteria ensure a thorough assessment of the algorithms' accuracy, robustness, computational efficiency, and overall applicability. This chapter outlines the metrics and conditions under which both algorithms were tested and compared. For the tables presented in this section, it is important to note that two experiments were performed for each output map determination: one for generating the output map and one for testing it. This procedure was conducted for both methods, and the reported measures (RMSE and R^2) represent the average of the two runs. Although the system operates in discrete time, continuous time has been used for the time axis in most plots for ease of interpretation and visualization.

7-1 Accuracy of Depth Estimation: Sonar-Based SLAM vs. SDP-GD

This section compares the depth estimation accuracy of the SDP-GD algorithm with that of the Sonar-Based SLAM approach. Both methods were tested under a series of experiments involving varying levels of seabed distortion. The distortion levels tested include the values $\{0, 1, 2, 3, 4, 6, 8, 10, 12\}$.

In each experiment, the AUV traveled approximately 100 meters along the x-direction. The collected data was subsequently downsampled to 6 Hz and truncated to 60 data points, representing a total distance of 10 meters.

Two distinct types of input signals were used: a reference input and a control-input. The reference input served as the reference signal and remained consistent throughout all experiments, thereby establishing a baseline for comparison. In contrast, the control-input was the actual input signal applied to the vehicle's thrusters. Due to environmental factors and disturbances, this signal varied between experiments. The reason behind using the control-input is that it contains more information regarding the dynamic interactions between the vehicle and its environment, thus allowing for a more thorough evaluation of the depth estimation capabilities under realistic and varied conditions.

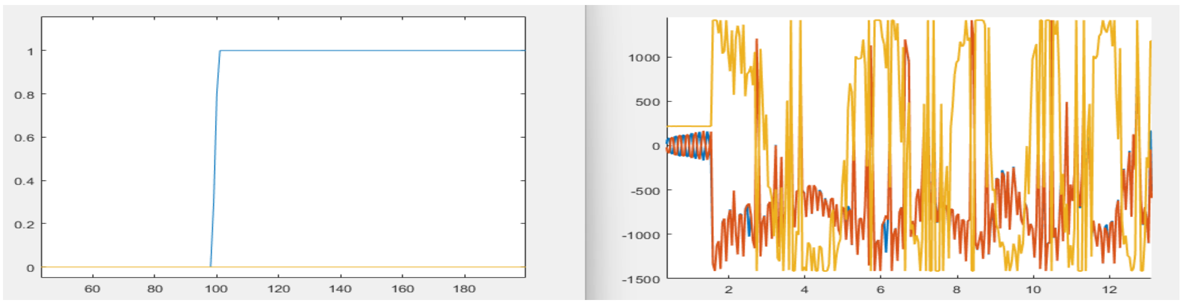


Figure 7-1: Comparison between reference and control-input signals. On the left, the reference input remains constant across all experiments. On the right, the control-input is influenced by environmental interactions, resulting in greater excitation of the system.

The goodness-of-fit metric used is the Root Mean Square Error (RMSE) of the output maps and the squared Pearson correlation coefficient.

7-1-1 Output map performance using Control Input

The output map performance using the control-input signal using different level of output noise will be compared in this section. Also different comparison are made includes both the SDP-GD and Sonar-Based SLAM algorithms across all distortion levels.

No output Noise

First the output map performance, made without any measurement noise is analysed. In both algorithm a $\Sigma_v = 0.0001$ is set, to prevent dividing by infinity. As shown in Table 7-1, Sonar SLAM outperforms SDP-GD in estimating the x-direction and depth, demonstrated by lower RMSE values and higher correlation coefficients for depth. In contrast, SDP-GD achieves better results in the y-direction, as indicated by a lower RMSE compared to Sonar SLAM.

For the reconstruction of the seabed map, the RMSE and R^2 Depth is used to analyse both algorithm performance. The result is plotted in fig be seen in fig 7-2 and its value is in table 7-1. The Sonar Based Slam algorithm outperforms the SDP-GD whenever no output noise is present. This can be seen by the higher RMSE and Lower R^2 levels, which indicates that the seabed reconstruction of the Sonar SLAM matches identically with the measurement. From the table it can be observed that the Sonar Based slam is better at estimating the pose of the AUV.

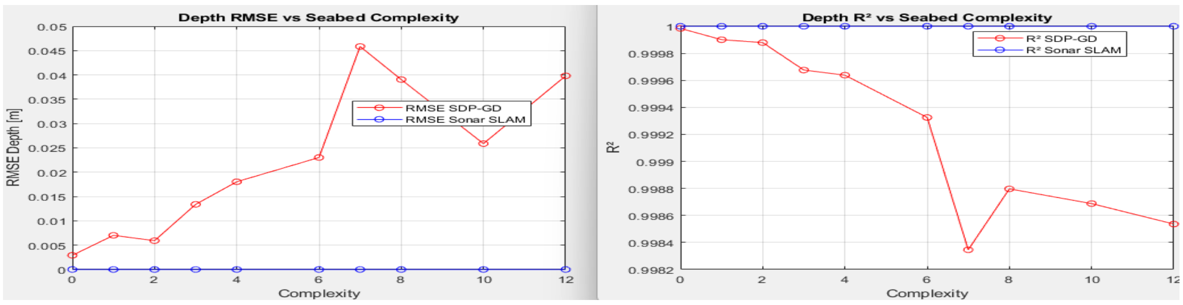


Figure 7-2: Comparison of the Average $RMSE$ and R^2 of the output map of the Sonar Based SLAM and SDP-GD, generated with measurement with no noise

Disturbance	RMSE SDP-GD (m)			RMSE SLAM (m)			SDP-GD	SLAM
	X	Y	Depth	X	Y	Depth	R_d^2	R_d^2
0	0.0183	0.0001	0.0068	0.0015	0.0060	0.0000	0.9999	1.0000
1	0.0150	0.0001	0.0056	0.0014	0.0060	0.0000	0.9998	1.0000
2	0.0253	0.0001	0.0129	0.0015	0.0066	0.0000	0.9991	1.0000
3	0.0149	0.0001	0.0075	0.0017	0.0060	0.0000	0.9998	1.0000
4	0.0249	0.0001	0.0087	0.0016	0.0066	0.0000	0.9999	1.0000
6	0.0150	0.0001	0.0057	0.0016	0.0059	0.0000	1.0000	1.0000
7	0.0044	0.0001	0.0022	0.0023	0.0045	0.0000	1.0000	1.0000
8	0.0125	0.0001	0.0069	0.0017	0.0050	0.0000	1.0000	1.0000
10	0.0278	0.0001	0.0109	0.0014	0.0066	0.0000	0.9998	1.0000
12	0.0052	0.0001	0.0015	0.0022	0.0047	0.0000	1.0000	1.0000

Table 7-1: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The input signal for both methods is the thruster input signal, which is used to determine the resulting output map. The measurement doesn't contain any noise

Measurement Noise = $\eta(0, 0.01)$, SNR = 53dB

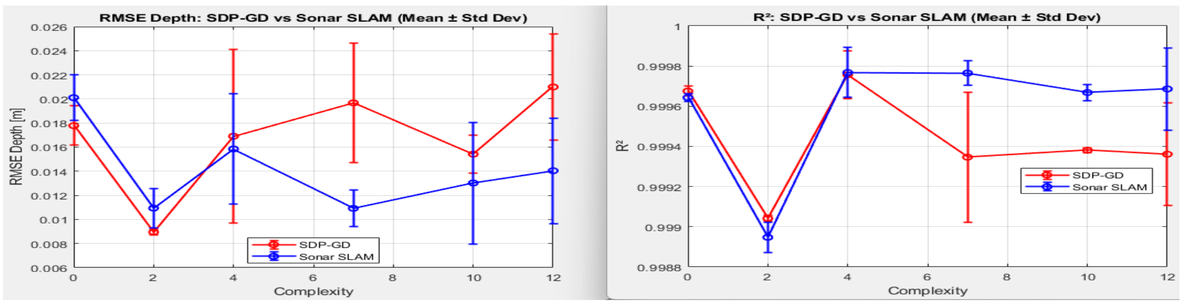


Figure 7-3: Comparison of the average $RMSE$ and R^2 of the output map generated by Sonar-Based SLAM and SDP-GD under measurement noise $\Sigma_v = 0.01$. The figure also includes the mean and standard deviation across multiple experimental runs.

Measurement Noise = $\eta(0, 0.1)$, SNR = 33dB

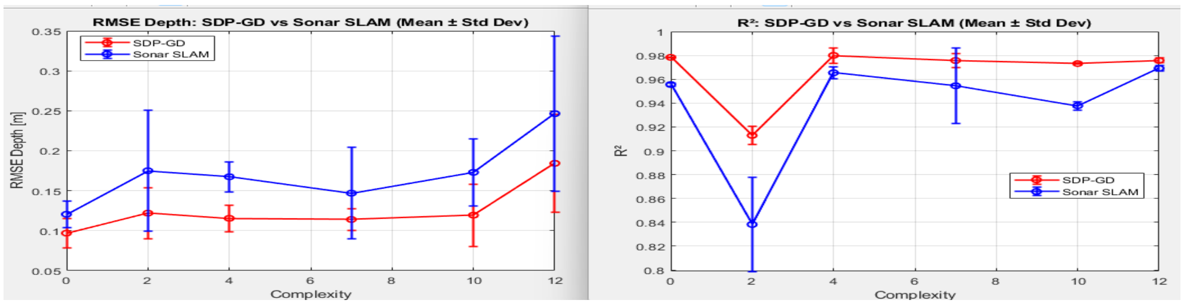


Figure 7-4: Comparison of the Average $RMSE$ and R^2 of the output map of the Sonar Based SLAM and SDP-GD, generated with measurement noise of $\Sigma_v = 0.1$. The figure also includes the mean and standard deviation across multiple experimental runs.

Measurement Noise = $\eta(0, 0.5)$, SNR = 19dB

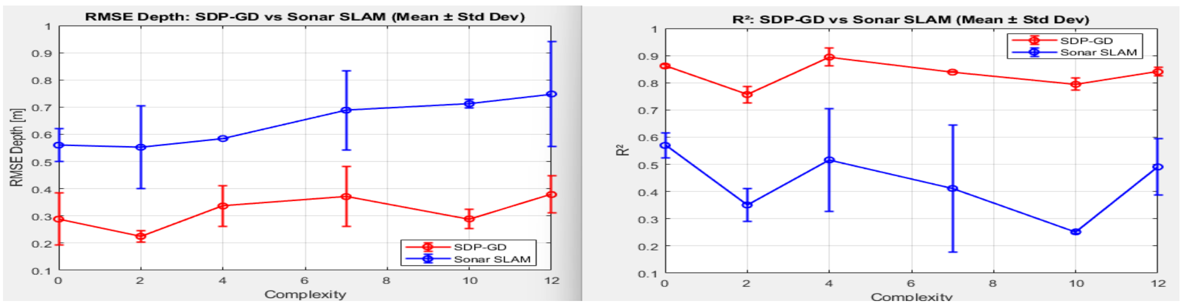


Figure 7-5: Comparison of the Average $RMSE$ and R^2 of the output map of the Sonar Based SLAM and SDP-GD, generated with measurement noise of $\Sigma_v = 0.5$. The figure also includes the mean and standard deviation across multiple experimental runs.

Measurement Noise = $\eta(0, 1.0)$, SNR = 13dB

In the previous results (Figures 7-3, 7-4, and 7-5), the measurement noise was progressively increased, corresponding to a lower Signal-to-Noise Ratio (SNR), and the output map was estimated using both SDP-GD and the Sonar-Based SLAM. Table 7-2 presents the results for a specific case where the SNR is 13 dB. Figure 7-7 plots the RMSE and R^2 values, demonstrating that SDP-GD consistently achieves a lower RMSE and higher R^2 under increased noise conditions. This suggests that SDP-GD is more robust to measurement noise compared to the Sonar-Based SLAM, which is also visually evident in Figures 7-3, 7-4, and 7-5, where SDP-GD begins to outperform Sonar-Based SLAM.

However, at very high noise levels, the R^2 drops significantly (e.g., $R^2 = 0.2$), indicating that the seabed reconstruction becomes increasingly unrecognizable from the measurements. This degradation in performance is further illustrated in Figure 7-6, where the RMSE is high and the R^2 reflects poor correlation with the true seabed.

An example where the seabed reconstruction is still recognizable is seen in figure 7-8.

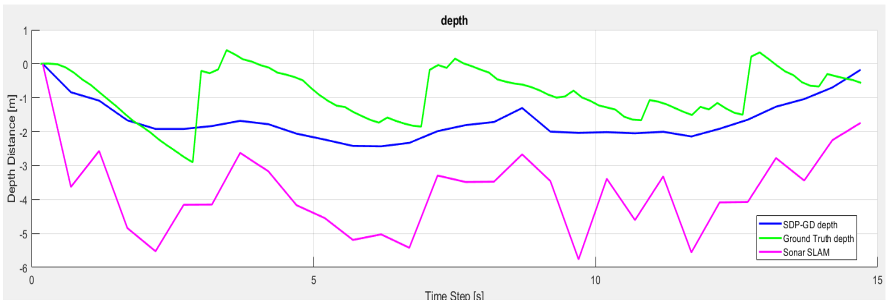


Figure 7-6: Example of the seabed reconstruction where the $R^2 = 0.2$ where the reconstruction is barely recognizable from the ground truth.

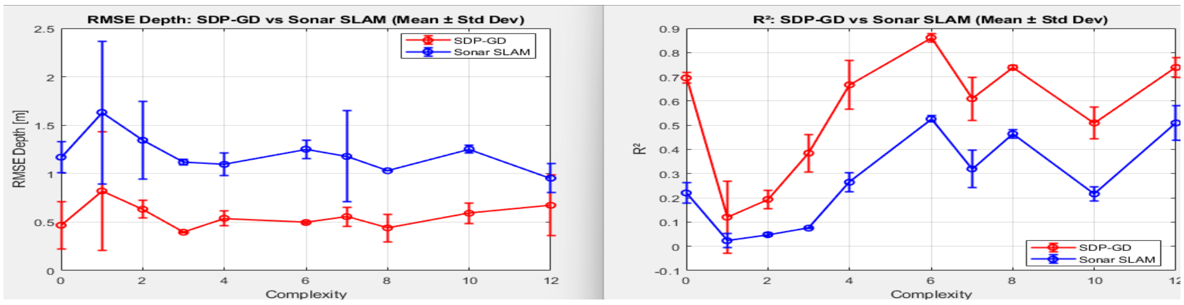


Figure 7-7: Comparison of the Average $RMSE$ and R^2 of the output map of the Sonar Based SLAM and SDP-GD, generated with measurement noise of $\Sigma_v = 1$.The figure also includes the mean and standard deviation across multiple experimental runs.

Disturbance	**RMSE SDP-GD (m)**			**RMSE SLAM (m)**			R_d^2	R_d^2
	X	**Y**	**Depth**	**X**	**Y**	**Depth**	**SDP-GD**	**SLAM**
0	0.0250	0.0129	0.4674	0.0626	0.0110	1.1693	0.6951	0.2204
1	0.0236	0.0109	0.8213	0.0612	0.0102	1.6323	0.1202	0.0233
2	0.0253	0.0127	0.6332	0.0628	0.0109	1.3447	0.1936	0.0481
3	0.0230	0.0110	0.3952	0.0633	0.0114	1.1182	0.3845	0.0752
4	0.0231	0.0112	0.5392	0.0619	0.0105	1.0954	0.6666	0.2647
6	0.0271	0.0144	0.4958	0.0625	0.0105	1.2526	0.8612	0.5257
7	0.0491	0.0269	0.5563	0.0729	0.0109	1.1779	0.6094	0.3197
8	0.0444	0.0231	0.4395	0.0729	0.0107	1.0299	0.7382	0.4644
10	0.0338	0.0133	0.5915	0.0627	0.0106	1.2533	0.5092	0.2164
12	0.0418	0.0196	0.6735	0.0731	0.0112	0.9531	0.7384	0.5092

Table 7-2: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.1)$ or a SNR = 13dB.

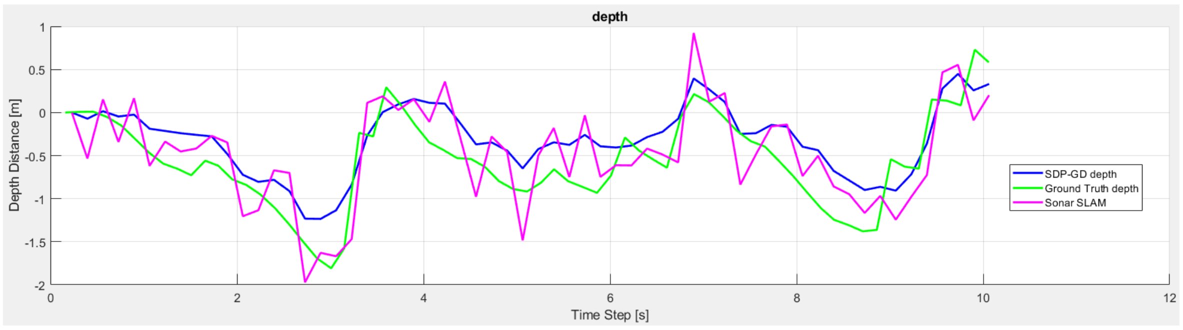


Figure 7-8: SDP-GD measures RMSE, R^2)=(0.32,0.82) and (0.37,0.623) for the Sonar SLAM. Here the output noise is equal to $\Sigma_v = 1$

7-1-2 Performance on reference signal Input C_{var1}

No Measurement Noise

In this subsection the reference signal is used as input signal. The same experimental setup is applied as in the control-input case, with the only difference being the altered input signal. The reasoning behind this is mostly to determine if an input signal with more variation will result in better or worse result.

When comparing the two algorithms directly under reference input wit no measurement noise, the results are similar to using a control input signal. The difference is that the RMSE are lower and the R^2 is higher.

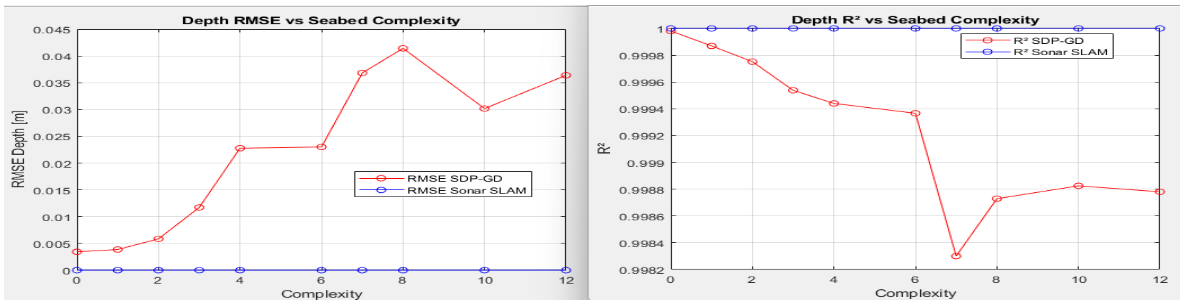


Figure 7-9: Comparison of the Average $RMSE$ and R^2 of the output map with no measurement noise using the Sonar Based SLAM and SDP-GD

When comparing the output map performance using both the control input and the reference input for each algorithm individually, it can be observed that using a reference input improves estimation accuracy in the y and depth directions for both algorithms, as shown in Table 7-3. In the following results, the measurement noise levels are increased from a SNR of 53dB to a SNR of 13dB.

Disturbance	RMSE SDP-GD (m)			RMSE SLAM (m)			SDP-GD	SLAM
	X	Y	Depth	X	Y	Depth	R_d^2	R_d^2
0	0.0042	0.0001	0.0010	0.0167	0.0005	0.0000	0.9999996	1.0000000
1	0.0042	0.0001	0.0005	0.0168	0.0005	0.0000	0.9999998	1.0000000
2	0.0042	0.0001	0.0006	0.0168	0.0005	0.0000	0.9999999	1.0000000
3	0.0043	0.0001	0.0007	0.0169	0.0005	0.0000	0.9999999	1.0000000
4	0.0042	0.0001	0.0008	0.0168	0.0005	0.0000	0.9999999	1.0000000
6	0.0042	0.0001	0.0005	0.0166	0.0005	0.0000	0.9999998	1.0000000
7	0.0038	0.0001	0.0006	0.0152	0.0004	0.0000	0.9999997	1.0000000
8	0.0038	0.0001	0.0007	0.0156	0.0004	0.0000	0.9999999	1.0000000
10	0.0043	0.0001	0.0005	0.0168	0.0005	0.0000	0.9999998	1.0000000
12	0.0038	0.0001	0.0006	0.0155	0.0004	0.0000	0.9999998	1.0000000

Table 7-3: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains no noise

Measurement Noise = $\eta(0, 0.01)$, SNR = 53dB

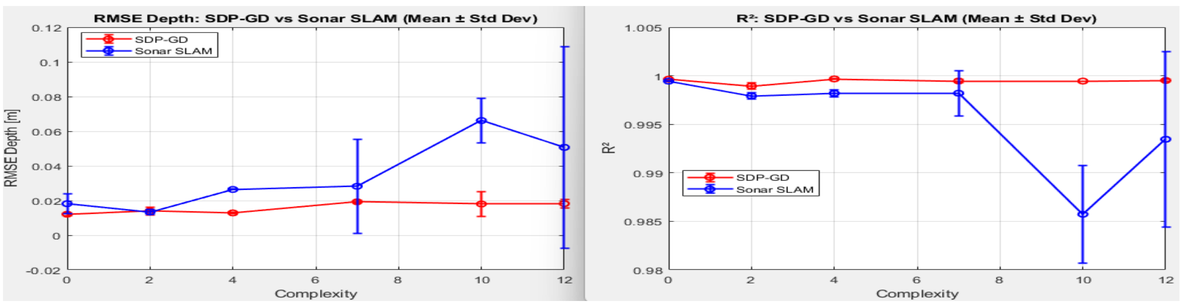


Figure 7-10: Comparison of the Average $RMSE$ and R^2 of the output map with measurement noise $\Sigma_v = 0.01$ of the Sonar Based SLAM and SDP-GD

Measurement Noise = $\eta(0, 0.5)$, SNR = 19dB

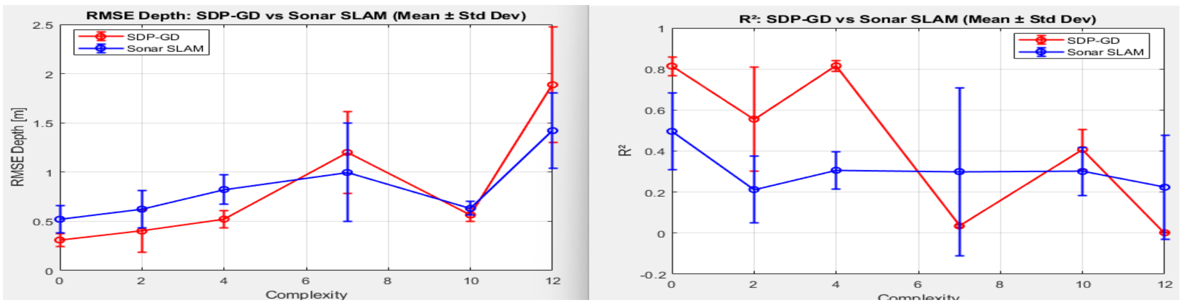


Figure 7-11: Comparison of the Average $RMSE$ and R^2 of the output map with measurement noise $\Sigma_v = 0.5$ of the Sonar Based SLAM and SDP-GD

Measurement Noise = $\eta(0, 1.0)$, SNR = 13dB

The effect of using a control input instead of a reference input signal can be observed in Figures 7-9, **??** and 7-14. These figures shows that SDP-GD outperforms Sonar-Based SLAM in reconstructing the seabed across all tested conditions—except in cases where no measurement noise is present. A similar performance trend is observed when the control input is used.

The results show that using the control input yields improved seabed reconstruction, as indicated by generally lower RMSE and higher R^2 values. This improvement is likely due to the control input capturing the true dynamics of the AUV more accurately, as it reflects the actual input used to drive the system, including real-time adjustments made during operation. Another, reason can be poor identification of the system model.

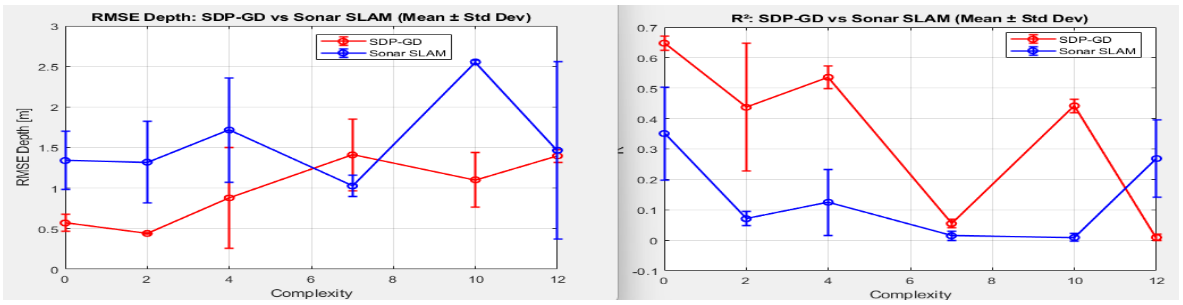


Figure 7-12: Comparison of the Average $RMSE$ and R^2 of the output map with measurement noise $\Sigma_v = 1$ of the Sonar Based SLAM and SDP-GD

Disturbance	RMSE SDP-GD (m)			RMSE SLAM (m)			R_d^2	R_d^2
	X	Y	Depth	X	Y	Depth	SDP-GD	SLAM
0	0.1149	0.0518	0.5701	0.0740	0.0060	1.3392	0.6477	0.3506
2	0.1745	0.0518	0.4410	0.0742	0.0057	1.3194	0.4369	0.0705
4	0.1813	0.0518	0.8794	0.0750	0.0061	1.7158	0.5355	0.1248
7	0.5944	1.0681	1.4090	0.0734	0.0049	1.0262	0.0547	0.0146
10	0.3487	0.0518	1.1019	0.0736	0.0063	2.5539	0.4416	0.0089
12	0.4723	1.0681	1.3968	0.0733	0.0047	1.4656	0.0092	0.2683

Table 7-4: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 1.0)$ or a SNR = 13dB.

From Table 7-4, Table 7-3, and the additional tables in the appendix showing output maps under varying measurement noise levels (Tables C-6 and C-7), it can be observed that increasing the measurement noise slightly degrades the position estimates of both the RMSE and the SLAM algorithm. This is indicated by the gradual increase in RMSE values as the measurement noise becomes higher. Additionally, the pose estimation appears to be more accurate when the control input is used as the input signal for the SDP-GD. This is reflected in lower RMSE values compared to those obtained when using the reference signal, suggesting that the control input enables a better representation of the system's true behavior.

In the case of Sonar-Based SLAM, the use of the control input also improves pose estimation accuracy. However, this comes at the cost of poorer seabed reconstruction, as indicated by generally higher RMSE values and lower R^2 scores. This suggests that while the pose estimate benefits from the control input, the associated trajectory may lead to less informative or more redundant sonar measurements, thereby reducing the quality of the reconstructed map.

This analysis shows that, regardless of the input signal used, SDP-GD consistently outperforms Sonar-Based SLAM in terms of pose estimation accuracy. This is mostly due to the nature of the SOnar based slam not containing a filtering option for the measurement noise. In the next section a filter is added to filter out the measurement noise.

7-1-3 Filtering Sonar Measurement

In the paper for the sonar based slam, no filter is being used to clean up the sonar side scan data. However, from the previous results it can be seen that the sonar data can be filtered. A 4th-order low-pass Butterworth filter is applied to the sonar measurement to remove high-frequency noise. The cutoff frequency is set to 2.5 Hz (from FFT analysis) and normalized with respect to the Nyquist frequency, which is half the sampling rate. Zero-phase filtering (filtfilt) is used to prevent phase distortion.

An example of using such filter can be seen in fig. 7-13, and the corresponding RMSE and R^2 plot can be seen in fig7-14.

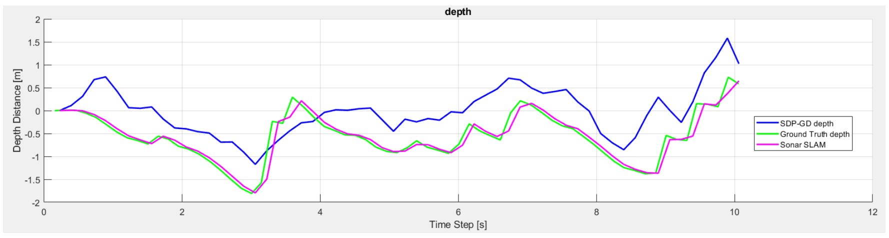


Figure 7-13: Example of seabed reconstruction at complexity level 10 using filtered sonar measurements. It is clearly observable that the Sonar-Based SLAM outperforms the SDP-GD in reconstruction accuracy.

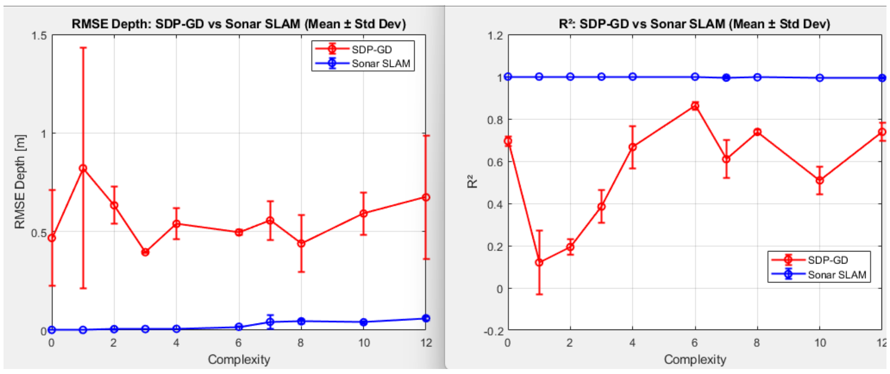


Figure 7-14: RMSE and R^2 performance comparison between SDP-GD and Sonar-Based SLAM across varying levels of seabed complexity, using filtered sonar measurements. The results clearly indicate that Sonar-Based SLAM outperforms SDP-GD under these conditions.

The application of this filter significantly improved the seabed reconstruction results for the Sonar-Based SLAM. However, this raises the question of fairness in comparison. While SDP-GD inherently includes a filtering mechanism (minimizing measurement error) as part of its formulation, Sonar-Based SLAM benefits from externally pre-filtered data. As a result, the comparison may be biased, since one algorithm performs its own filtering, whereas the other relies on a separate preprocessing step.

7-1-4 Computational Efficiency

This section compares the computational efficiency of the SDP-GD and Sonar-Based SLAM algorithms by evaluating the time it took to compute the output maps under increasing data volumes. In these experiments, multi-threading was used. The data is down-sampled to 2 Hz, and a time horizon n_T of 30, 60 and 100 data points are compared. Computations were executed using MATLAB's `parfor` (parallel for-loop) function, which enables parallel processing of independent tasks. In this case, the three output maps, corresponding to the x, y, and depth direction, were computed independently and later merged. This separation made parallelization possible.

Distortion	**30 Samples**		**60 Samples**		**100 Samples**	
	SDP-GD (s)	**SLAM (s)**	**SDP-GD (s)**	**SLAM (s)**	**SDP-GD (s)**	**SLAM (s)**
0	38.6104	0.0002	406.3856	0.0004	1219.2000	0.0012
1	40.1605	0.0002	391.7984	0.0004	—	—
2	40.1610	0.0002	396.8178	0.0003	—	—
3	39.9548	0.0002	378.5677	0.0004	—	—
4	40.8118	0.0002	372.7263	0.0004	—	—
6	40.4914	0.0002	376.7389	0.0004	—	—
8	23.4219	0.0002	364.1409	0.0004	—	—
10	41.5364	0.0002	366.9418	0.0004	—	—
12	6.7675	0.0002	237.5535	0.0004	—	—

Table 7-5: Comparison of average computational time (in seconds) for the SDP-GD and Sonar-Based SLAM algorithms across different seabed distortion levels and sample sizes.

Based on the values shown in Table 7-5, it can be observed that the computational time of the SDP-GD algorithm varies across experiments. This variation can be attributed to the nature of the solver used in SDP-GD, which in some cases converges more quickly to an optimal solution depending on the structure of the input data or the optimization landscape. In addition to variation across experiments with the same number of samples, the computational time of the SDP-GD algorithm also increases significantly as the number of samples grows. For reference, the computation of a single output map using 100 samples with SDP-GD took approximately 20 minutes to complete. For example, using 30 samples, the computation time reached up to 41 seconds, while for 100 samples, a single output map took approximately 20 minutes to compute.

In contrast, the Sonar SLAM algorithm employs a Kalman filter, which follows a recursive estimation process with a predictable and fixed computational cost per time step. As a result, its runtime remains relatively consistent regardless of the input or environmental complexity.

The sharp increase in computation time for SDP-GD as the number of samples grows can be explained by its computational complexity, as described in [1]:

$$\mathcal{O}\left(\frac{n_x^3(n_T+1)^3+n_x^2(n_T+1)^2}{2}+n_yn_x^2(n_T+1)^2+n_xn_y^2(n_T+1)^3+n_y^3(n_T+1)^3\right). \quad (7\text{-}1)$$

This expression simplifies to:

$$\mathcal{O}(n_T^3) \quad \text{when } n_x, n_y \ll n_T,$$

where n_x is the number of states, n_y the number of outputs, and n_T the discrete time horizon. This shows that the computational time increases with the power of 3 with respect to the time horizon. Therefore, increasing the number of samples, either by extending the sampling duration or by using a higher sampling rate, leads to a significant increase in computation time for SDP-GD, while the SLAM approach remains unaffected.

7-1-5 Robustness to Environmental Conditions

This section evaluates the robustness of the SDP-GD and Sonar SLAM algorithms under varying water current conditions. A 6 Hz sampling rate was used in simulations to capture environmental noise, although real-world applications would likely require higher rates to account for higher-frequency disturbances.

Figures 7-15 and 7-16 compare SDP-GD and Sonar SLAM trajectories under low (0.2 m/s) and high (1.2 m/s) current levels, respectively, at seabed complexity level 12. The green line indicates ground truth, the dotted magenta line shows the nominal state, the solid magenta line is the Sonar SLAM estimate, and the blue line represents SDP-GD results (which is perfectly behind the SLAM Signals).

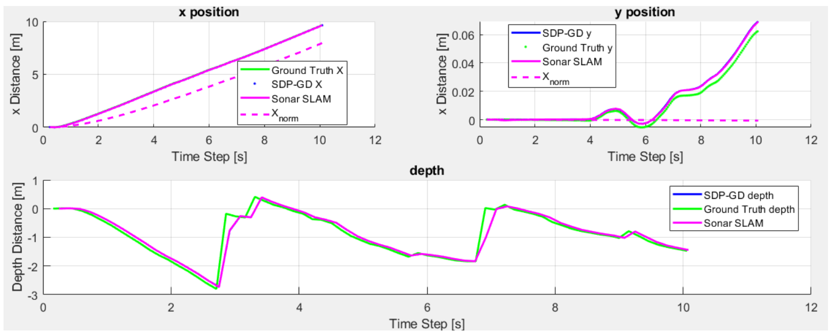


Figure 7-15: Response of SDP-GD and Sonar SLAM in low-current (0.2 m/s) conditions at high seabed complexity. Sampling rate: 2 Hz, horizon: $n_T = 60$.

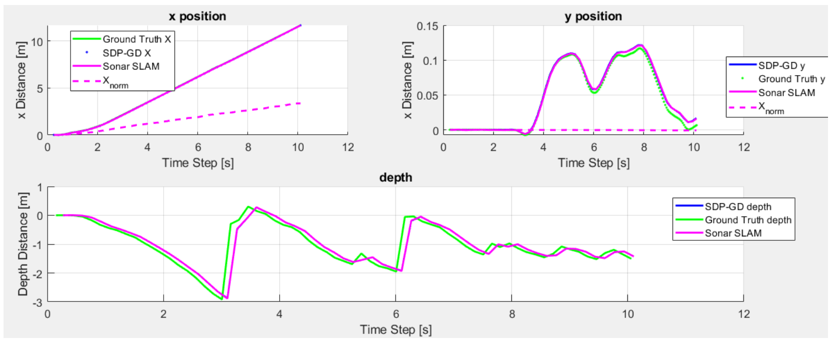


Figure 7-16: Response of SDP-GD and Sonar SLAM in high-current (1.2 m/s) conditions at high seabed complexity. Sampling rate: 2 Hz, horizon: $n_T = 60$.

From the analysis, it can be observed that environmental conditions have no significant impact on the accuracy of pose estimation and seabed reconstruction. This conclusion is based on a model identified under conditions without current-induced noise, serving as a baseline. The magenta line (X_{norm}), which represents the expected trajectory, doesn't aligns with the reconstructed path which indicates the robustness even against change in path. As a result, both SDP-GD and Sonar-Based SLAM demonstrate strong robustness to environmental disturbances such as water currents.

7-1-6 Ease of Integration and Use

However, significant challenges arose during the implementation of the lifted observation model. In particular:

- **Noise Covariance and Observation Structure:** Configuring the observation model to properly handle noise covariances proved non-trivial. Errors often only became apparent at the optimizer level, manifesting as constraint failures that provided little insight into the underlying issues.
- **Parameter Adjustment and Structure Flexibility:** The default implementation assumed a $1 \times n_C$ observation matrix structure. Extending this to a $3 \times n_C$ configuration required concatenating three separate observation maps. Since zeroing out specific components was not directly supported, a workaround was needed by artificially inflating the corresponding noise variance to effectively ignore undesired observations.
- **Terminology Inconsistencies:** The code's variable names, such as `nominalX`, did not match the terminology used in the associated paper. Clarifying these terms (e.g., recognizing that `nominalX` corresponds to $A.u$ in the system model) would have reduced the learning curve and eased debugging.

In summary, although the core algorithm remains well-structured, enhancements in flexibility, consistent terminology, and debugging support—especially regarding the observation model—would significantly improve its usability and adaptability across different systems.

7-2 ORB-SLAM2 vs. SDP-GD

After generating the seabed reconstructions using both SDP-GD and ORB-SLAM2, the outputs can be directly compared. While previous sections evaluated both x- and y-coordinates, for ORB-SLAM2 only the depth component was analyzed due to time constraints.
Figure 6-3 shows a visual comparison of the reconstructions, with the ground truth obtained from a noise-free sonar sensor. Mean Squared Error (MSE) and Pearson correlation coefficients were computed across two runs per trajectory (described in Section 6-4), and results are summarized in Table 7-6.
Overall, SDP-GD outperforms ORB-SLAM2 in multiple cases, achieving higher Pearson correlations and often lower MSE values.
For Trajectory 1, the SDP-GD Pearson correlation declines and MSE increases with rising map complexity, reflecting the algorithm's sensitivity to environmental variations. In contrast, results for Trajectories 2, 3, and 4 appear more arbitrary across complexity levels. This inconsistency stems from the maps used—scaled versions of a relatively flat environment—lacking the diverse terrain features necessary for robust evaluation.
Trajectory 1 includes moderate hills and grooves, offering more complexity compared to the others, and results indicate that SDP-GD performs better than ORB-SLAM2 under increased terrain variability.
While additional experiments with more varied seabed geometries and repeated runs are needed for conclusive findings, these preliminary results suggest that SDP-GD provides more accurate seabed reconstructions, especially when fusing measurements from both ORB-SLAM2 and the sonar-based system (second variation).

Table 7-6: Comparison of MSE and Pearson Correlation for SDP-GD and ORB-SLAM2 using sonar as the measurement. The trajectories (1–4) are explained in Section 6-4, and the complexities in Section 3-3.

Complexity	1	2	3	4	Avg
MSE SDP-GD					
0	0.15626	0.40343	0.32785	0.14546	0.25875
1	0.51269	0.51597	14.43600	0.03210	3.37469
3	0.96317	0.71138	1.22510	0.45882	0.83962
MSE ORB-SLAM2					
0	0.16252	0.36934	0.47866	0.09014	0.27517
1	0.45460	2.28530	1.68460	0.76338	1.29647
3	1.02100	0.74821	4.05300	2.02280	1.96175
Pearson Correlation SDP-GD					
0	0.96175	0.88607	0.89521	0.91638	0.91485
1	0.96398	0.91956	0.40083	0.99690	0.82082
3	0.90901	0.85956	0.87839	0.91790	0.89172
Pearson Correlation ORB-SLAM2					
0	0.97470	0.88803	0.88424	0.96118	0.92704
1	0.95552	0.51393	0.89148	0.89527	0.81405
3	0.86888	0.84874	0.80215	0.58085	0.77516

7-2-1 Effect of Murky Water on Seabed Reconstruction

In this section, the effect of murky water conditions on seabed reconstruction is analyzed. Figure 7-17 illustrates the output of the SDP-GD algorithm (yellow) , ORB-SLAM2 (blue) , and the ground truth measured by the sonar (red). The data used here simulates a murky underwater environment where the Sonar SLAM misinterprets suspended particles as the seabed, causing incorrect orientation estimations, as previously shown in Figure 3-9.

It can be observed that the SDP-GD reconstruction incorrectly estimates the seabed as rising, whereas the actual seabed remains level. This highlights a limitation of the current SDP-GD implementation, which heavily relies on the AUV's positional data. In noisy or visually degraded environments, such as murky water, this dependency can lead to inaccurate reconstructions. This presents a significant challenge for real-world applications, as many underwater environments are not clear.

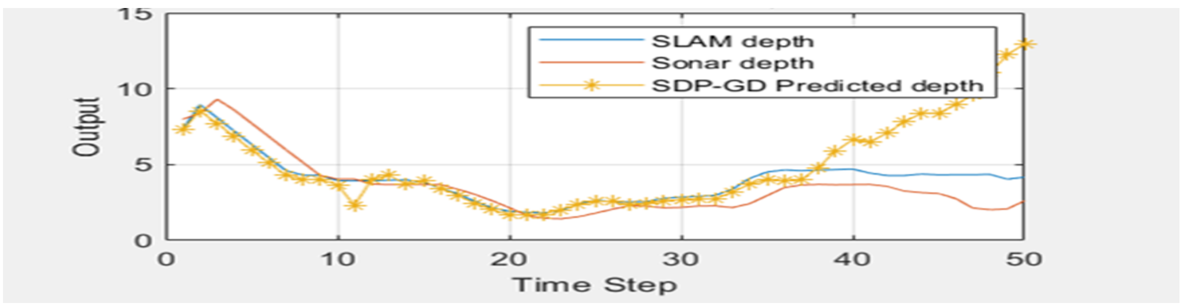


Figure 7-17: Seabed reconstruction under murky water conditions using SDP-GD (blue), ORB-SLAM2 (blue), and sonar-based ground truth (red).

Chapter 8

Discussion and Future Work:

8-1 Discussion the Simulation Setup

Impact of Sparse Visual Features on SLAM Evaluation: The simple simulated environment used in this study limited the visual SLAM system's ability to detect reliable keypoints, affecting its performance compared to SDP-GD. To enable fairer comparisons in future work, environments with more distinct visual features (e.g., rocks, corals) should be used. Reducing visual noise, employing feature detectors optimized for underwater conditions, and incorporating stereo or RGB-D sensors could further enhance visual input quality and enable more accurate depth estimation, improving both SLAM evaluation and 3D reconstruction comparisons.

Current Modeling Constraint: The simulation modeled currents only in the x-axis direction (facing the AUV), limiting disturbance evaluation. Future work should develop a controller supporting multidirectional motion to better assess the impact of currents from different angles.

Control Input: In the comparison, the control input with relatively high frequency components—was used. However, these high-frequency components may have been a bit too high, which likely introduced issues during the down-sampling process. This limitation was only identified at a later stage of the analysis. Using an input signal with higher bandwidth requires higher sampling rates to prevent aliasing. However, if the objective is to minimize computational time, such high-frequency inputs should be avoided, as they increase the computational burden without necessarily improving estimation performance.

Identification: A first-order linear model was used for BlueROV2 system identification due to time constraints. While sufficient for straight-line motion, future work should use richer input signals to capture higher-order dynamics for broader applicability.

Computational Efficiency of SDP-GD: To reduce computational time in the SDP-GD algorithm in the visual SLAM comparison, batching was introduced in the third variant of the SDP-GD implementation. This approach proved effective when using a time horizon of $nT = 1$, and resulted in only a marginal increase in computation time. Future improvements may involve experimenting with different batch sizes to extend the time horizon without sacrificing computational efficiency. This could make it feasible to apply the SDP-GD framework in more complex or real-time scenarios.

Accurate Data Synchronization: The accuracy of seabed reconstruction was found to be sensitive to the timing of the visual SLAM measurements, especially at the onset of descending and forward motion. In the experiment, the recorded time window was slightly shorter than required, leading to an inaccurate seabed estimation. If the timing of the measurements

had been precise, this error could likely have been avoided.
For future research, it is important to ensure accurate synchronization between the model's motion and the data collection process, especially when determining the start of key maneuvers.

8-2 Evaluation of Experimental Results

Variation of Input Signal: In this part of the thesis, the AUV is simulated in different seabed complexities and different measurement noise.

Computational Efficiency and Sampling Frequencies: SDP-GD is significantly more computationally intensive than Sonar SLAM due to its reliance on a solver. Increasing the number of states or extending the time horizon raises computational cost. In this study, the system was simplified to six states (x, y, z, v_x, v_y, v_z) to match straight-line AUV motion without rotation.
In more general applications, especially with full 6-DOF motion, the state vector may need to be expanded to 12 states, including rotational terms. Given the complexity scales as $\mathcal{O}(n_T^3)$, reducing the time horizon n_T is key to improving efficiency.
The sampling rate directly impacts n_T. Here, data was downsampled to 6 Hz, which helped reduce computation. Higher sampling rates retain more motion detail and improve estimation accuracy by increasing the number of time steps within a fixed duration.

Robustness to Environmental Conditions: In the section "Robustness to Environmental Conditions", the AUV was simulated under varying underwater current disturbances. Despite increasing the environmental noise (e.g., water currents), the performance of the output map remained largely unaffected. This consistent accuracy indicates that both algorithms—SDP-GD and Sonar-Based SLAM—are robust against environmental disturbances. This robustness is likely attributed to the presence of a feedback loop in both methods, which continuously corrects the estimated state using incoming measurements, thereby minimizing estimation errors.

8-3 Bayesian4Wiener

Recently, S. Vakili et al. published a new paper titled *Optimal Bayesian Affine Estimator and Active Learning for the Wiener Model* [26], which presents an updated method that builds upon the SDP-GD framework (Bayesian4Wiener algorithm). This section discusses the improvements introduced in that work and how they address limitations identified in this thesis.

Faster Estimation: The current SDP-GD approach requires solving a non-convex optimization problem, initialized via a semidefinite program (SDP). While effective, this process becomes computationally demanding for large-scale systems or long time horizons. In contrast, the new method avoids numerical optimization altogether during the estimation phase. Instead, it computes the posterior distribution directly, resulting in significantly faster performance—particularly for static nonlinear maps.

Nonlinear Output Map: Another key advancement is the introduction of a nonlinear output map. The current method in this thesis uses a linear output map, which is suitable given that the simulated system dynamics are relatively linear (e.g., straight-line motion). However, in real-world scenarios, AUVs often perform complex maneuvers involving nonlinear behaviors. As a result, the mapping from the AUV's pose to the seabed reconstruction can also become nonlinear, which highlights the usefulness of employing a nonlinear output map in such scenarios.

Active learning In the current SDP-GD framework, the algorithm does not adapt or improve with different inputs, where its learning capacity is fixed once the initial model is identified.

In contrast, the updated method incorporates active learning, where control actions—such as oscillating, diving, or altering its trajectory— enables the ability to actively explore the environment.

8-4 Conclusion

It is important to note that only one component of the Sonar-Based SLAM algorithm was implemented in this study, forming just one part of a more comprehensive system. This component creates a single sub map, the other component includes patching multiple sub-maps to create a global map.

This thesis demonstrates that SDP-GD is comparable to Sonar-Based SLAM in generating sub-maps. However, the computational time required by SDP-GD is significantly higher. This is largely due to its dependence on the sampling rate and time horizon—factors that must be carefully balanced. In contrast, Sonar-Based SLAM is considerably faster in computing the output map.

A limitation of Sonar-Based SLAM is that it does not explicitly account for noise in the sonar measurements. This is due to the unkown relationship between the state and the sonar side-scan measurements. One possible workaround is to introduce a preprocessing step that filters out noise from the sonar data before it is used in SLAM. Such filtered measurements could also be integrated into the SDP-GD framework to enhance the accuracy of the output map.

The analysis shows that both Sonar-Based SLAM and SDP-GD yield comparable results in seabed reconstruction. This raises a critical question regarding the necessity of the significantly longer computation time required by SDP-GD—especially given that Sonar-Based SLAM achieves similar performance much faster, particularly when a filtering step is applied to the sonar measurements.

The newly proposed framework, *Bayesian4Wiener*, has the potential to overcome several limitations of the SDP-GD approach presented in this thesis. Its implementation is recommended for future research, particularly in scenarios requiring faster inference, nonlinear modeling capabilities, and active learning strategies. The method can be tested with online SLAM algorithms.

Appendix A

Appendix: Simulation Setup

A-1 Gazebo plugins used

Hydrodynamic and Control Plugins

For the design of an AUV, hydrodynamic phenomena have to be taken into account, as these influence the vehicle's motion and stability in underwater environments. Such phenomena include flow patterns, wave dynamics, pressure effects, and buoyancy. In the Gazebo simulation, these physical effects are modeled using a combination of specialized plugins. This subsection discusses the key plugins used to simulate buoyancy, hydrodynamic forces, and thruster control.

The Buoyancy Plugin is responsible for calculating and applying buoyant forces to each link in the AUV model [27]. These forces are applied at the center of each link and depend on the fluid density surrounding the object. The center of volume can either be explicitly specified or, if left undefined, automatically computed based on the link's geometry. Given the mass and volume of each link, the plugin uses Archimedes' principle to determine the net buoyancy force, allowing the AUV to behave realistically in the simulated water column.

The Hydrodynamics Plugin models forces experienced by bodies underwater due to drag, lift, and inertial coupling with the surrounding fluid [28]. This plugin accounts for the added mass induced by the inertia of displaced water, the Coriolis-centripetal matrix resulting from this added mass, and a hydrodynamic damping model. The damping includes components such as radiation-induced potential damping, skin friction, wave drift damping, vortex shedding, and lifting forces. Together, these forces are captured in a hydrodynamic damping matrix. The plugin requires the ground truth state of the vehicle and environment as input and includes parameters for added mass, linear drag, and quadratic drag across all six degrees of freedom: surge, sway, heave, roll, pitch, and yaw.

The Thruster Plugin simulates maritime thrusters typically used for underwater propulsion [29]. This plugin can be configured to accept either a desired thrust force (in Newtons) or an angular velocity input (ω) for each thruster. The parameters required for configuring the plugin include the fluid density (ρ), the propeller diameter (D), and the thrust coefficient (C_T), which relates angular velocity to thrust, which is positive for clockwise rotation and negative for counter-clockwise. The plugin also supports velocity control modes, where setting `velocity_control` to `true` directly specifies the rotational speed of the propeller. When set to `false`, a PID controller is used to apply wrenches that achieve the desired motion. Additional parameters include the proportional (P), integral (I), and derivative (D) gains for control, maximum and minimum allowable thrust or angular velocity, and the minimum effective input below which the thruster produces no meaningful output. A wake fraction parameter is also included to model the decrease in speed of water near the propeller relative to the free-stream flow.

The angular velocity ω required to achieve a given thrust can be calculated using the following relation:

$$\omega = \sqrt{\frac{\text{thrust}}{\rho \cdot C_T \cdot D^4}} \tag{A-1}$$

Additionally, the advance speed V_a, adjusted for wake effects, is given by:

$$V_a = (1 - \text{wake\_fraction}) \cdot \text{advance\_speed} \tag{A-2}$$

Together, these plugins form the core of the underwater simulation, allowing the AUV to interact with its virtual environment in a physically realistic manner.

Coordinate Frames and Transforms

The coordinate system used in the simulation follows the REP-105 standard [30], which defines several reference frames commonly used in robotic systems. In this context, the simulation makes use of the body-fixed frame, and two world-fixed frames: odom and map. These frames are hierarchically connected through a series of static and dynamic transforms, which allow for accurate tracking of the AUV's position and orientation in both local and global contexts.

The body-fixed frame defines a coordinate system attached to the AUV itself. The origin is typically located at the geometric center of the vehicle, and the axes are aligned with the structure of the AUV as illustrated in Figure 3-1b. This frame is essential for describing motion in the AUV's own reference, such as control inputs or IMU readings, and it remains constant relative to the vehicle.

The world frame "odom" is a local, inertial frame that remains fixed relative to the AUV's initial position at the start of the simulation. According to REP-105, this frame is allowed to drift over time without bounds, making it unsuitable for long-term global localization. However, it provides smooth and continuous tracking of the AUV's pose, which is ideal for velocity estimation and local trajectory planning. In the simulation setup, the initial position is defined as $[0, 0, 0]$, with the AUV facing east. The origin of the z-axis is located at the surface of the water.

The world frame "map" serves as the global reference frame. Like the odom frame, its initial pose is also set to $[0, 0, 0]$. Unlike odom, however, the map frame is designed not to drift significantly over time. It can undergo discrete updates, or "jumps," allowing for correction of accumulated drift in localization. This makes the map frame particularly useful for tasks that require consistent long-term positioning, such as loop closure in SLAM systems.

Transforms between frames are established through both static and dynamic relationships. Static transforms are used for fixed elements in the system, such as from the camera frame to the base frame, or from the base frame to the sensor frame, since these components are physically mounted in known, unchanging positions on the AUV. Dynamic transforms, on the other hand, represent the changing relationships during simulation. The transform from `map` to `odom` captures how the local odometry drifts relative to the global frame, and is continuously updated as the AUV navigates the environment. Meanwhile, the transform from `odom` to `base_link` tracks the AUV's position and orientation over time relative to its starting point. Together, these transforms allow the AUV's position to be consistently expressed across different coordinate systems, which is essential for both control and evaluation tasks.

A-2 BlueROV2 Configuration

The BlueROV2, developed by BlueRobotics, is used as the AUV for the simulation. It is available in two configurations: the base model and the heavy configuration.

The two configurations differ primarily in their thruster setup and payload capacity. The base configuration includes six thrusters: four for forward/reverse movement and yaw control, and two vertical thrusters for depth control. The heavy configuration expands this with eight thrusters by adding two additional vertical thrusters. These enhance vertical stability and control. Furthermore, the heavy variant includes a buoyancy system designed for heavier payloads and offers a higher depth rating. A schematic of the heavy configuration is shown in Figure A-1, while Figure 3-1 provides a visualization of the BlueROV2 in the Gazebo simulation environment, including its axis orientation.

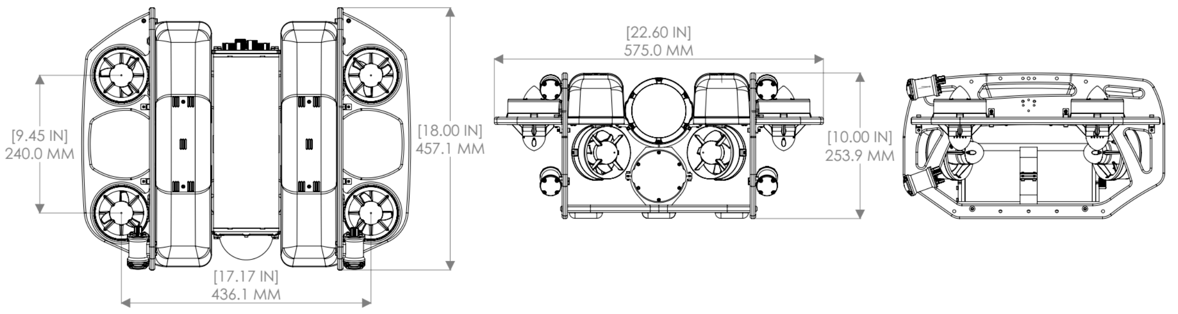


Figure A-1: BlueROV2 Heavy configuration 2D schematic [31]

Table A-1: BlueROV2 parameters

Parameter	Symbol	Value	Unit
Length	L	0.457	m
Width	W	0.338	m
Height	H	0.254	m
Mass	m	10.565	kg
Pitch moment of inertia	I_{zz}	0.201	kg·m
Surge added mass	I_{Ax}	10.565	kg
Heave added mass	I_{Ay}	10.565	kg
Pitch added mass	I_{An}	0.201	kg·m
Quadratic damping coefficient	C_D	0.5	-
Surge cross-sectional area	A_x	0.048	m^2
Sway cross-sectional area	A_y	0.10	m^2
Yaw cross-sectional area	A_n	0.07	m^2

Detailed specifications for both configurations can be found on the official BlueRobotics website [17], [31].

A-3 ORB-SLAM2

Loop closing

The process of detecting and closing loops in the path or trajectory of a system as it moves through an environment, is called *loop closure* [32]. Newer SLAM algorithms include loop closure to prevent perceiving the environment as an infinite corridor. A simple example of how a map built by odometry and SLAM differ is shown in fig. A-2, from [33]. In this example, a robot is mapping an environment, starting from point A and aiming for point B. While exploring, the robot goes to point C but misses point B and later reaches point B through another path. Without using loop closures, point B and C are assumed to be far away, while in reality they are close together as shown by the map built by SLAM. SLAM estimates the location using the environment structure and using loop closures, and by doing so, the algorithm can detect shortcuts (going from B to C).

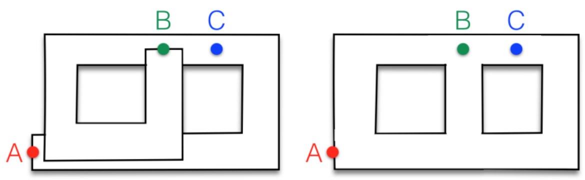


Figure A-2: Map built by odometry (left) vs SLAM (right). According to the left map, points B and C are distant from each other, while in reality they are supposed to be adjacent to each other as on the map built by SLAM. By estimating based on the environment structure, SLAM can identify shortcuts within the map by using loop closures. Adapted from C.Cadena, et. al. [33]

The loop closing process identifies significant loops and corrects accumulated drift by performing pose-graph optimization.

Tracking failure

Whenever tracking fails, the algorithm is equipped with a placed recognition module based on DBoW2 [34]. This module is useful for realization in already mapped scene, loop detection and occlusion.

Keypoint Detection and System Initialization

The initial stage of the ORB-SLAM2 pipeline involves pre-processing of stereo image inputs to extract visual features for tracking and mapping. This process is illustrated in Figure A-3. For feature detection, the system uses ORB (Oriented FAST and Rotated BRIEF) features [35], which are designed to be invariant to scale and rotation, and robust against changes in auto-gain, auto-exposure, and illumination. These properties make them well-suited for challenging environments such as underwater settings.

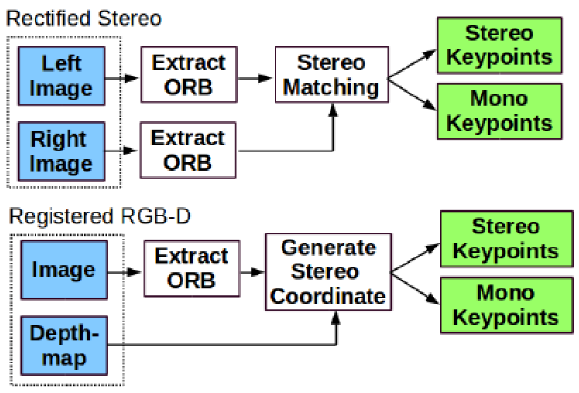


Figure A-3: ORB architecture taken from [19]

The system employs a stereo vision setup in which two identical cameras are mounted in a fixed, parallel configuration. This ensures that the stereo images are rectified. This means that corresponding points in the left and right images lie on the same horizontal line. If a point is observed at pixel coordinates $x_L = (H_L, V_L)$ in the left image and $x_R = (H_R, V_R)$ in the right image, rectification ensures $V_L = V_R$. In this setup, keypoints can be represented as $x_s = (H_L, V_L, H_R)$, where (H_L, V_L) are the coordinates in the left image, and H_R is the horizontal coordinate in the right image.

Keypoints are classified into two categories: **close** and **far** keypoints. According to the definition in [36], a keypoint is considered close if its depth is less than 40 times the stereo camera baseline, which is the distance between the two camera centers. Close keypoints can be triangulated with high accuracy from a single frame, resulting in reliable estimates for scale, translation, and rotation. In contrast, far keypoints provide accurate orientation information but offer less precise scale and translation estimates. A visual example of keypoint detection and their mapping in the simulation is shown in Figure A-5.

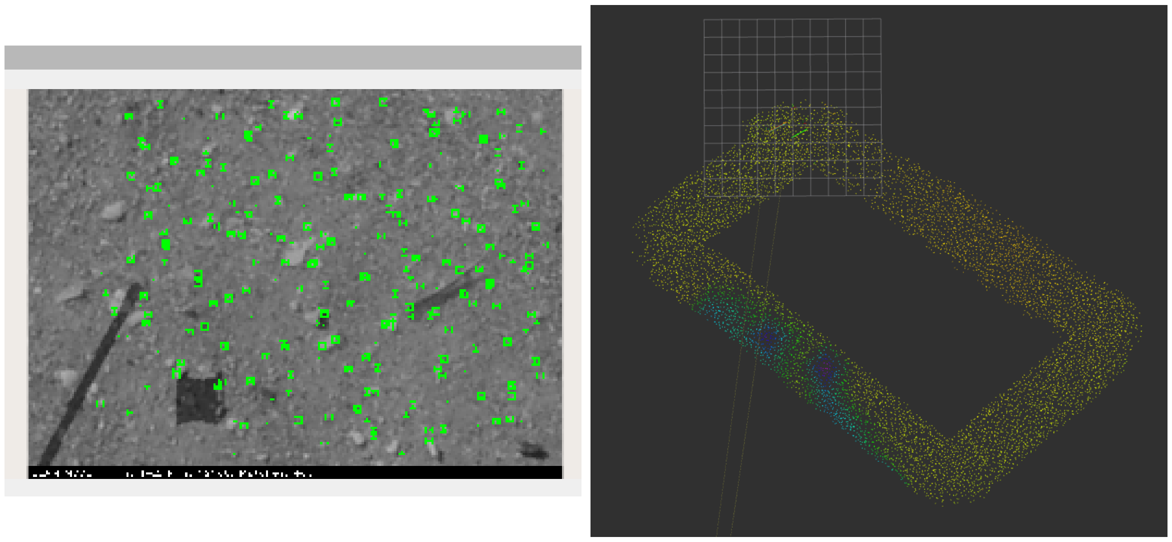


Figure A-5: Trajectory with the keypoints in Gazebo

Due to the availability of stereo depth information, ORB-SLAM2 can initialize the system with a single stereo frame. The initial set of keypoints and corresponding map points are established based on the AUV's starting pose, which is assumed to be at the origin $(0, 0, 0)$. This bootstrapping step provides a reliable foundation for subsequent tracking and mapping operations.

Bundle Adjustment

Once keypoints have been detected and matched, the camera pose and the corresponding 3D map points are refined using an optimization technique known as *Bundle Adjustment (BA)*. This process minimizes the *reprojection error*, which is defined as the difference between the observed image points and the reprojected locations of the estimated 3D points.

Bundle Adjustment can be formulated as a non-linear least-squares optimization problem [37]. The objective is to optimize the 3D structure of the map and the camera poses by minimizing the sum of reprojection errors across all observations. This minimization is typically performed using iterative least-squares solvers, with the Levenberg–Marquardt algorithm being a commonly used method due to its balance between convergence speed and stability.

A-4 SLAM-Algorithm: Sonar Based SLAM

Sensors

Sonar: The sonar sensor used for the simulator is the **Teledyne BlueView P900**. This Sonar sensor has the following specification:

Table A-2: Specifications of the Teledyne BlueView P900 2D Imaging Sonar [38]

Specification	**Value**
Operating Frequency	900 kHz
Update Rate	Up to 15 Hz
Maximum Range	100 m (328 ft)
Optimum Range	2–60 m (6.5–197 ft)
Beam Width	1° x 20°
Beam Width	1° x 20°
Field-of-View	45°,90°, 130°

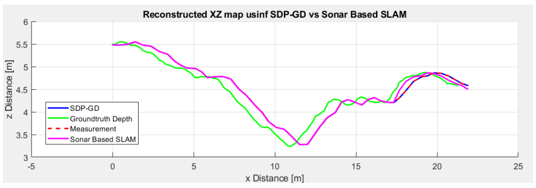
Figure A-6: Teledyne BlueView P900: 2D Imaging Sonar [38]

IMU: The IMU used for the simulator is the ADIS16448 of Analog Devices.

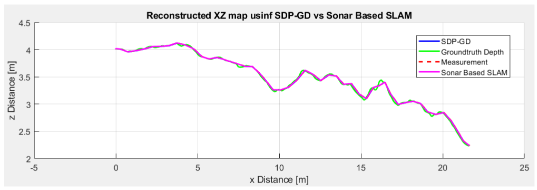
Table A-3: Specifications of the Analog Devices ADIS16448 IMU [39]

Specification	**Value**
Gyroscope Range	$\pm 250°$/sec, $\pm 500°$/sec, $\pm 1000°$/sec
Accelerometer Range	± 18 g minimum
Magnetometer Range	± 1.9 gauss minimum
Barometer Operating Range	10 mbar to 1200 mbar
Calibrated Pressure Range	300 mbar to 1100 mbar
Start-Up Time	205 ms
Operating Temperature Range	-40°C to 105°C
Supply Voltage	3.15 V to 3.45 V
Shock Survivability	2000 g
Dimensions	24.1 mm $\times$ 37.7 mm $\times$ 10.8 mm

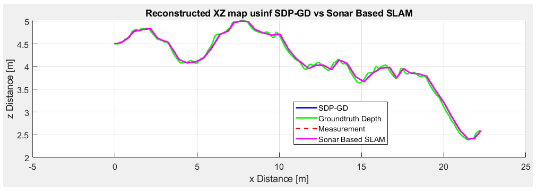
A-5 Different seabed profiles, with their distortion levels



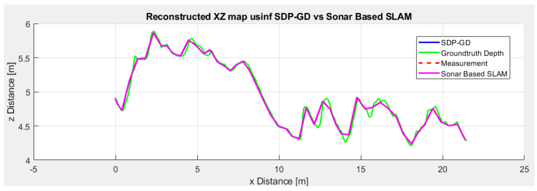
(a) Distortion level 0



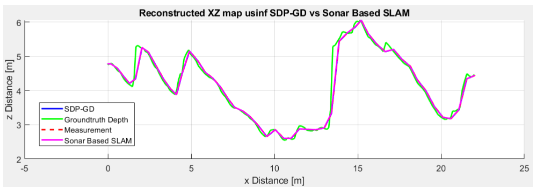
(b) Distortion level 1



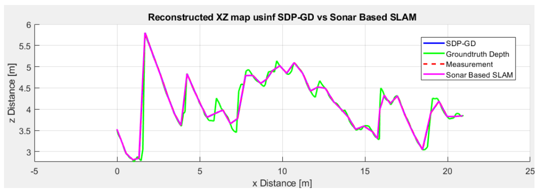
(c) Distortion level 2



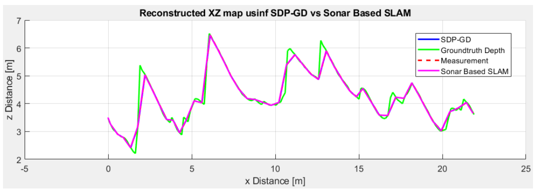
(d) Distortion level 3



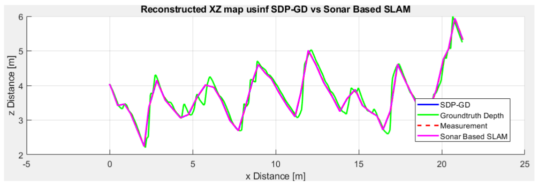
(e) Distortion level 4



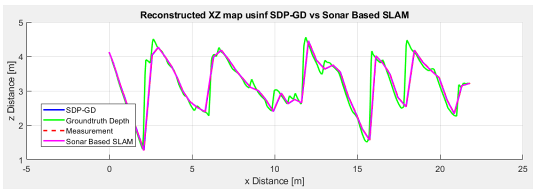
(f) Distortion level 6



(g) Distortion level 8



(h) Distortion level 10



(i) Distortion level 12

Figure A-7: Comparison of SDP-GD and Sonar-Based SLAM under increasing distortion levels.

A-6 Model Components

Model Components

Each term in the dynamic model introduced in eq. 5-1 plays a specific role in representing the motion and interactions of the AUV with its environment. This section describes each of these components in detail.

Coordinate Transformation Matrix $J(\eta)$

In eq. 5-1, $J(\eta)$ represents the transformation matrix that maps velocities from the body-fixed frame to the world frame. It is block-diagonal and consists of translational and rotational transformation components:

$$J(\eta) = \begin{bmatrix} J_1(\eta) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & J_2(\eta) \end{bmatrix} \tag{A-3}$$

The submatrices $J_1(\eta)$ and $J_2(\eta)$ are defined as:

$$J_1(\eta) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\cos\phi\sin\theta \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{A-4}$$

$$J_2(\eta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{A-5}$$

Note: $J_2(\eta)$ is not defined for $\theta = \pm\frac{\pi}{2} + k\pi$, where $k \in \mathbb{Z}$, due to singularities in the tangent and secant functions. This phenomenon is known as *gimbal lock*

Inertia Matrix M

In eq. 5-1, M represents the strictly positive definite inertia matrix, which is composed of two parts: the rigid-body inertia matrix M_{RB} and the hydrodynamic added-mass matrix M_A:

$$M = M_{RB} + M_A \tag{A-6}$$

The rigid-body inertia matrix M_{RB} and added-mass matrix M_A are defined as:

$$\begin{aligned} M_{RB} &= \begin{bmatrix} m\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & I_C \end{bmatrix}, \\ M_A &= -\mathrm{diag}\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\} \end{aligned} \tag{A-7}$$

Here, $I_C = \mathrm{diag}\{I_x, I_y, I_z\}$ is the moment of inertia tensor about the principal axes. The terms $\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\}$ represent the hydrodynamic added-mass forces and moments along the corresponding degrees of freedom in surge, sway, heave, roll, pitch, and yaw, respectively.

Coriolis Matrix $C(\nu)$

The skew-symmetric Coriolis matrix $C(\nu)$ from eq. 5-1 accounts for the effects of rotational and translational motion on the dynamics of the AUV. It consists of two components: one due to the rigid-body motion and one due to the added mass effects:

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) \tag{A-8}$$

where:

- $C_{RB}(\nu)$: the rigid-body Coriolis-centripetal matrix, representing the fictitious forces arising from the AUV's own rotational and translational motion,
- $C_A(\nu)$: the added-mass Coriolis matrix, which models the corresponding fictitious forces resulting from the hydrodynamic added mass (i.e., the virtual mass of water moving with the vehicle).

$$C_{RB} = \begin{bmatrix} 0 & 0 & 0 & 0 & -mv & mu \\ 0 & 0 & 0 & mv & 0 & -mu \\ 0 & 0 & 0 & -mu & mv & 0 \\ 0 & -mv & mu & 0 & I_{zz}r & -I_{yy}q \\ mv & 0 & -mu & -I_{zz}r & 0 & I_{xx}p \\ -mu & mv & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \tag{A-9}$$

$$C_A = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{A-10}$$

The values are simulator-specific, and actual numerical forms are used during implementation.

Hydrodynamic Damping Matrix $D(\nu)$

In eq. 5-1, the hydrodynamic damping matrix $D(\nu)$ represents resistive forces acting on the AUV due to water drag. It consists of a linear damping component D, and a nonlinear damping component $D_n(\nu)$, such that:

$$D(\nu) = D + D_n(\nu) \tag{A-11}$$

At the low speeds considered in this study, the AUV is assumed to experience uncoupled damping, as confirmed by the controller results. This allows the damping matrices to be modeled as diagonal, with no coupling between degrees of freedom [40].

The linear damping matrix D is given by:

$$D = -\text{diag}\{X_u, Y_v, Z_w, K_p, M_q, N_r\} \tag{A-12}$$

The nonlinear damping matrix $D_n(\nu)$, which accounts for quadratic drag effects, is expressed as:

$$D_n(\nu) = -\text{diag}\{X_{|u|u}|u|, Y_{|v|v}|v|, Z_{|w|w}|w|, K_{|p|p}|p|, M_{|q|q}|q|, N_{|r|r}|r|\} \tag{A-13}$$

In the simulator used for this study, only nonlinear damping coefficients are included.

Gravitational and Buoyancy Forces and Moments $g(\eta)$

The gravitational and buoyancy forces in eq. 5-1 can be expressed as:

$$g(\eta) = \begin{bmatrix} (W-B)\sin\theta \\ -(W-B)\cos\theta\sin\phi \\ -(W-B)\cos\theta\cos\phi \\ y_b B\cos\theta\cos\phi - z_b B\cos\theta\sin\phi \\ -z_b B\sin\theta - x_b B\cos\theta\cos\phi \\ x_b B\cos\theta\sin\phi + y_b B\sin\theta \end{bmatrix}, \tag{A-14}$$

where (x_b, y_b, z_b) are the coordinates of the center of buoyancy in the body frame. The gravitational force W and buoyancy force B are defined as:

$$\begin{aligned} W &= mg, \\ B &= \rho g \Delta. \end{aligned} \tag{A-15}$$

The buoyancy forces B depend on the water's density (ρ) and the volume of the fluid that was displaced by the vehicle (Δ).

External Forces and Moments τ

In eq. 5-1, external forces are present along every axis on the AUV:

$$\tau = \begin{bmatrix} f \\ r \times f \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ F_z l_y - F_y l_z \\ F_x l_z - F_z l_x \\ F_y l_x - F_x l_y \end{bmatrix} = T(\alpha)F = T(\alpha)Ku \tag{A-16}$$

A-7 Disturbance Analysis: Current effect on the Desistek SAGA Rov

X-Direction

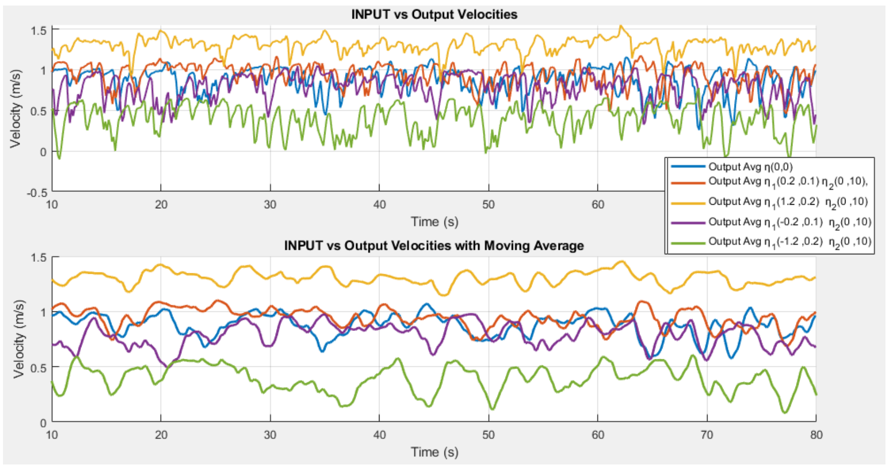


Figure A-8: AUV Velocity response of the ROV (down) under different simulated current velocities along the x-axis (top). Ignore the titles!!.

It can be observed that varying current velocities have a nonlinear effect on the velocity of the AUV. Despite this, the position of the ROV continues to increase approximately linearly in the x-direction.

Y-Direction

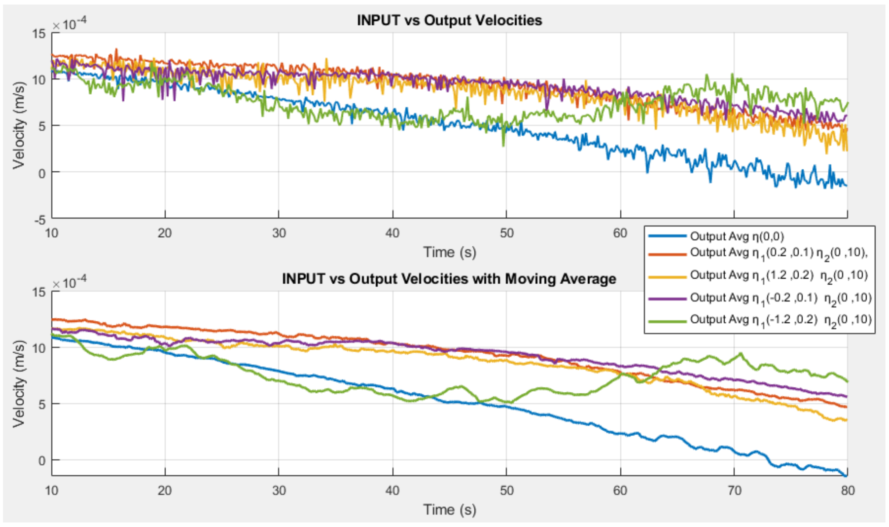


Figure A-9: AUV Velocity response of the ROV (down) under different simulated current velocities along the y-axis (top). Ignore the titles!!.

When the ROV is facing the current, the current has little effect on the y-direction at lower velocities. However, at higher current velocities, the ROV begins to exhibit noticeable oscillations in the y-direction.

Z-Direction

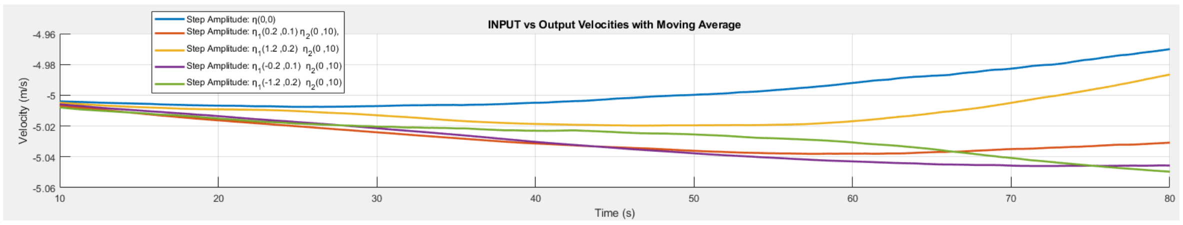


Figure A-10: Ground truth position of the ROV under varying simulated current velocities in the z-direction.

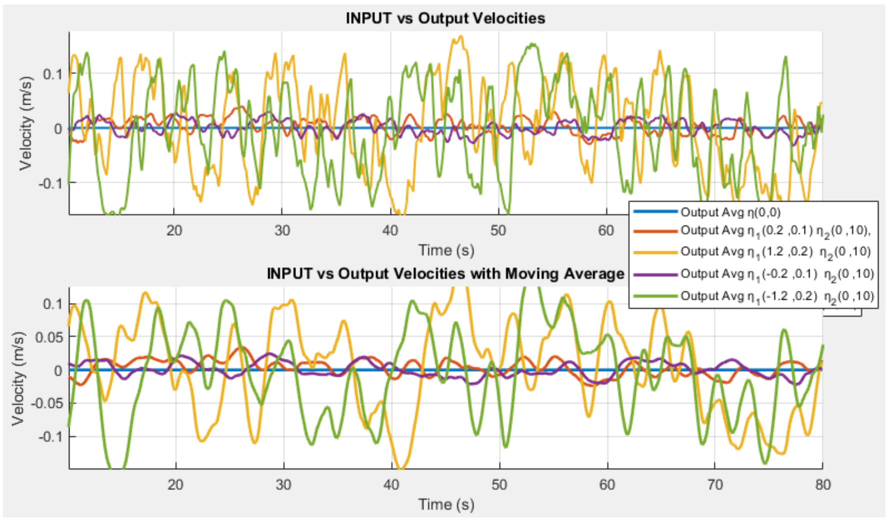


Figure A-11: Ground truth position of the ROV under varying simulated current velocities in the z-direction.

The varying current velocity does not appear to significantly affect the ROV's z-velocity or z-position. This is primarily because the current does not vary in the vertical direction, only in the horizontal direction.

Mean Velocity Comparison

The table below presents the average velocity observed in each axis under different current conditions, while maintaining a reference velocity of $u_{\text{ref}} = (u_x, u_y, u_z) = (1, 0, 0)$.

Table A-4: Effect of Current on Mean Velocity in Each Axis

Parameter	x	y	z
$\eta_1(1.2, 0.2)\ \eta_2(0, 10)$	1.30	8.602×10^{-4}	0.0033
$\eta_1(0.2, 0.1)\ \eta_2(0, 10)$	0.94	9.289×10^{-4}	0.0032
$\eta(0, 0)$	0.87	5.140×10^{-4}	0.0001
$\eta_1(-0.2, 0.1)\ \eta_2(0, 10)$	0.79	9.243×10^{-4}	0.0007
$\eta_1(-1.2, 0.2)\ \eta_2(0, 10)$	0.39	7.514×10^{-4}	-0.0038

Here it can be notice that the error in the y velocity is minimal but in the figure A-9 it can be noticed that the effect

Appendix B

Appendix: Data Pre-Processing

B-1 ORB-SLAM2 Pre-Processing

Data Gathering

Prior to use of the data the data requires to be processed to in readable form. The raw data has the following plot whenever going straight for 100 meters at a depth of -6.5 meters. The output ma

Data Preprocessing and Filtering

Interpolation

Both signals are sampled using different frequencies, therefore the samples are different. The SLAM algorithm has the lowest amount of samples, therefore the system states are down sampled to the size of the input signal.

Removing Outliers

The measurement contains measurement noise, which are infinite. These values are removed from the data using a threshold.

Delays

The signals starts of on the same magnitude, however a delay is present in the data. To account for this, rather than adding data (padding) all data is cut whenever it reaches a certain magnitude, which is set to 9.6.

Reference Values and Initial Conditions

Both trajectories are handles as if they were 2 different dataset, therefore 2 intial conditions are set for each dataset. The data is cut at $t_{\text{cutoff}} = 25$ seconds. This time is chosen based on the time the AUV reaches the depth of 6.5 meters. This time will differ based on different t_{cutoff}.

Different reference values and initial conditions are applied before and after t_{cutoff}:

- Before t_{cutoff}: reference $= (0, 0, -5)$, $x_0 = [0, 0, 0]$
- After t_{cutoff}: reference $= (100, 0, -5)$, $x_0 = [0, 0, -5]$

Transformation matrix

Pre-Processing

Nearest-Neighbor

A *KD-tree* as constructed on the ground truth map for efficient *nearest-neighbor lookups*, which enables quick error calculations during the alignment process.
The nearest-neighbor uses a point from the SLAM point cloud ($\mathbf{P}_i$) to determine the corresponding nearest point ($\mathbf{Q}_i$):

$$\mathbf{Q}_i^* = \arg\min_{\mathbf{Q}_j \in \mathbf{Q}} d(\mathbf{P}_i, \mathbf{Q}_j) \tag{B-1}$$

$$d(\mathbf{P_i}, \mathbf{Q}_j) = \sqrt{(p_x - q_{jx})^2 + (p_y - q_{jy})^2 + (p_z - q_{jz})^2} \tag{B-2}$$

Time sync

s The first step is to synchronize the data. This is done by checking whenever the signals reaches the threshold 6, which the data is cut from this point.

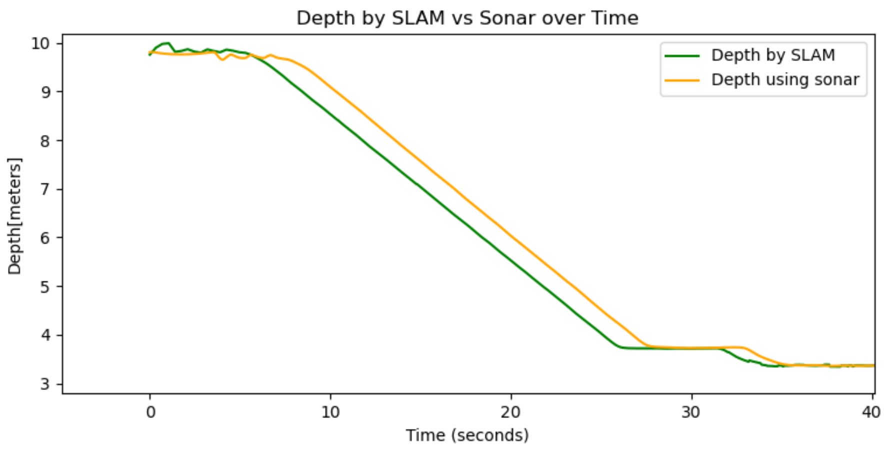


Figure B-1: Enter Caption

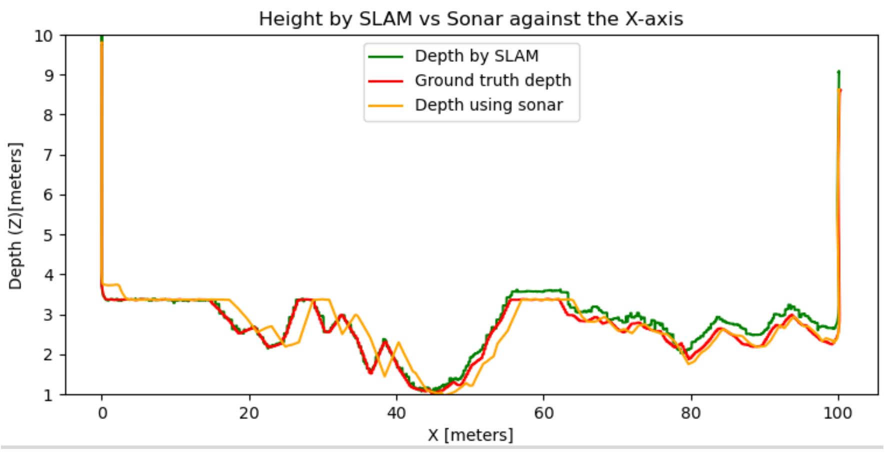


Figure B-2: Enter Caption

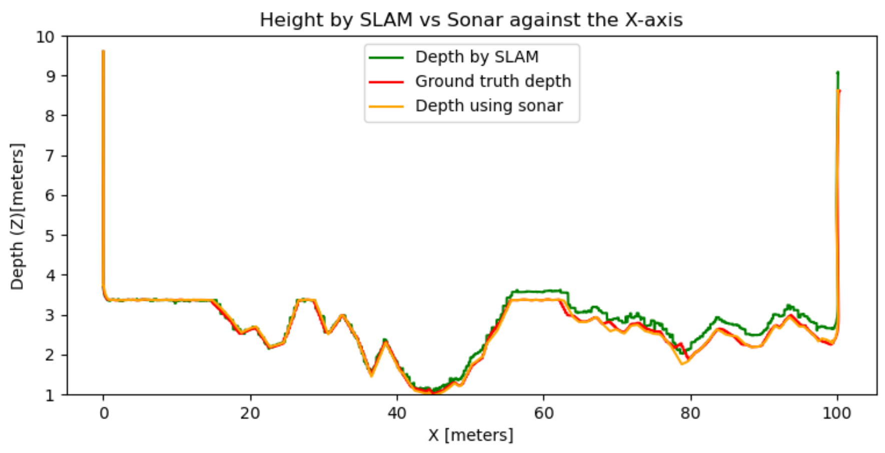


Figure B-3: Enter Caption

Down-sampling

After applying the bias and scale, the samples of the 3 signals requires to be set equal to each other.

Table B-1: Sampling Rates

Description	**Sampling Rate (Hz)**
Lidar sampling rate	20.00
SLAM map sampling rate	4.74
SLAM location sampling rate	17.98
Ground truth sampling rate	50.00

In order to compare datapoints, the ground truth data and lidar data are downsampled to the size of the SLAM data points size. This is due to the SLAM location having the lowest sampling rate, which means lower data points.

Result

Error

The error of both lidar and SLAM can be seen decreasing the lower the AUV is of the ground.

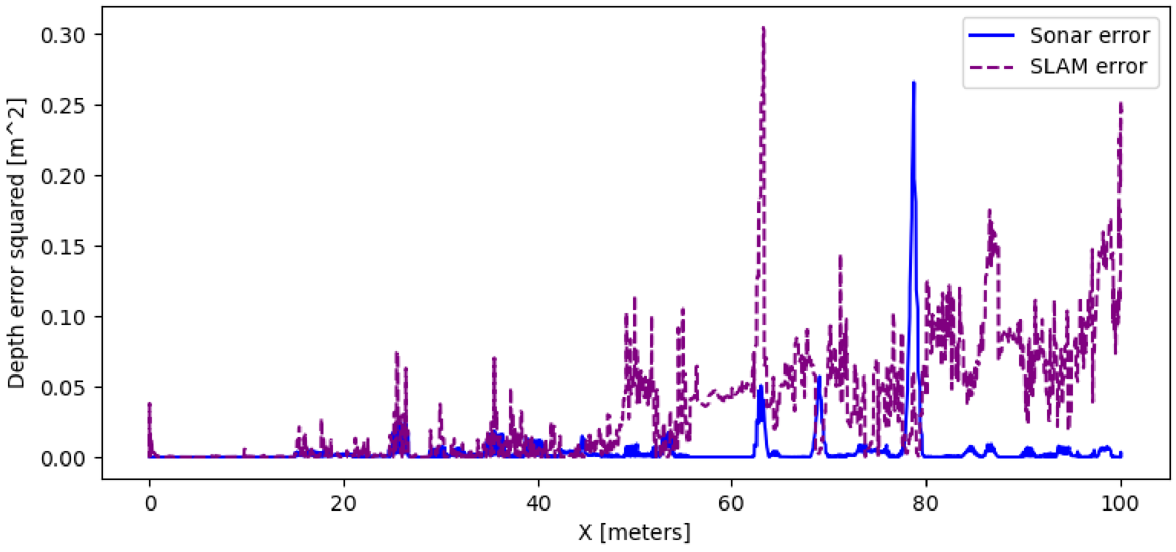


Figure B-4: Enter Caption

Table B-2: Mean Squared and Variance of Lidar and SLAM Errors for Different Depths

Depth (m)	**Time Interval (s)**	**Mean Squared Lidar Error**	**Variance Lidar Error**	**Mean Squared SLAM Error**	**Variance SLAM Error**
-6.0	6–11	0.00018	0.00018	0.00148	0.000042
-7.0	15–20	0.00331	0.00016	0.00441	0.000050
-8.0	24.5–29	0.01132	0.00119	0.00534	0.000191
-9.0	33.5–38.5	0.02206	0.00414	0.00776	0.000199

B-2 Identification results

Table B-3: Model Fit (%) for Identification Using Chirp(0,1) and Chirp(1,0.5)

	Chirp(0,1)		Chirp(1,0.5)	
Output	Grey-Box Fit (%)	Black-Box Fit (%)	Grey-Box Fit (%)	Black-Box Fit (%)
y_1	14.42	-88.25	99.16	98.26
y_2	0.02	-74.33	82.09	71.68
y_3	1.97	-123.62	33.23	33.14
y_4	14.54	17.89	17.70	15.15
y_5	-74.07	-114.20	21.11	19.25
y_6	1.63	1.34	-0.22	6.07

Table B-4: Model Fit (%) for Validation Using Chirp(0,1) and Chirp(1,0.5)

	Chirp(0,1)		Chirp(1,0.5)	
Output	Grey-Box Fit (%)	Black-Box Fit (%)	Grey-Box Fit (%)	Black-Box Fit (%)
y_1	28.07	-6.87	98.70	97.62
y_2	-484.44	-38605.49	-2087.93	-1094.86
y_3	-10662.11	-71464.18	-880.76	-869.66
y_4	-577.45	-2022.22	44.28	45.05
y_5	-40166.68	-83054.43	-4705.91	-1072.59
y_6	-46125.48	-265647.71	-1342.20	-3495.92

Appendix C

SDP-GD implementation and evaluation

Variation 3 for Sonar Based SLAM: $C_{\textsf{var}_3}$

In this variation, two output maps (C_1 and C_2) are used to determine the location of the AUV and the corresponding depth profile. The first output map, denoted as C_1, uses the thruster input signals (as in Variation 2) and takes the x and y positions as measurements:

$$y_{\text{meas}_1} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{C-1}$$

The optimized output from the first map, y_{C_1}, is then used as the nominal state x_{nominal} in the second output map. This nominal state, derived from the Lifted Observation Model, is defined as:

$$x_{\text{nominal}} = C_1(\theta_1)Au \tag{C-2}$$

Since x_{nominal} is already computed, the second output map does not require any new input signals. The measurement for the second output map is the depth profile, which corresponds to the x and y positions:

$$y_{\text{meas}_2} = \text{depth}$$

The predicted depth is then computed as:

$$\hat{y}_{\text{meas}_2} = C_2 x_{\text{nominal}} = C_2(\theta_2)C_1(\theta_1)Au \tag{C-3}$$

Using an initial condition $(x[k], y[k]) = (0, 0)$, according to eq C-3 can lead to a a initial estimation error whenever the depth is non zero. This results in an incorrect depth prediction. To compensate for this, a bias is introduced to shift the estimated signal.

C-1 Output map performance using Control Input

Measurement Noise = $\eta(0, 0.01)$, 53dB

Disturbance	**RMSE SDP-GD (m)**			**RMSE SLAM (m)**			R_d^2	R_d^2
	X	**Y**	**Depth**	**X**	**Y**	**Depth**	**SDP-GD**	**SLAM**
0	0.0008	0.0005	0.0178	0.0021	0.0007	0.0201	0.9997	0.9996
2	0.0008	0.0005	0.0089	0.0021	0.0007	0.0109	0.9990	0.9989
4	0.0008	0.0005	0.0169	0.0021	0.0007	0.0158	0.9998	0.9998
7	0.0030	0.0012	0.0197	0.0123	0.0023	0.0109	0.9993	0.9998
10	0.0021	0.0006	0.0154	0.0021	0.0006	0.0130	0.9994	0.9997
12	0.0014	0.0007	0.0210	0.0123	0.0023	0.0140	0.9994	0.9997

Table C-1: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.01)$ or a SNR = 53dB.

Measurement Noise = $\eta(0, 0.1)$, 33dB

Table C-2: RMSE and R^2 values for SDP-GD and Sonar SLAM across disturbance levels

Dist.	**SDP-GD**			**Sonar SLAM**			R^2 SDP-GD	R^2 S
	RMSE X	RMSE Y	RMSE Depth	RMSE X	RMSE Y	RMSE Depth		
0	0.0070	0.0044	0.0969	0.0058	0.0022	0.1205	0.9783	0.9
2	0.0063	0.0040	0.1219	0.0059	0.0022	0.1750	0.9129	0.8
4	0.0060	0.0041	0.1155	0.0060	0.0020	0.1674	0.9800	0.9
7	0.0180	0.0092	0.1140	0.0240	0.0040	0.1469	0.9757	0.9
10	0.0112	0.0045	0.1194	0.0059	0.0019	0.1730	0.9731	0.9
12	0.0121	0.0059	0.1843	0.0240	0.0043	0.2464	0.9757	0.9

Table C-3: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.1)$ or a SNR = 33dB.

Measurement Noise = $\eta(0, 0.5)$, 19dB

Table C-4: RMSE and R^2 values for SDP-GD and Sonar SLAM across disturbance levels

Dist.	SDP-GD			Sonar SLAM			R^2 SDP-GD	R^2 S
	RMSE X	RMSE Y	RMSE Depth	RMSE X	RMSE Y	RMSE Depth		
0	0.0175	0.0100	0.2882	0.0315	0.0069	0.5601	0.8618	0.5
2	0.0171	0.0095	0.2248	0.0315	0.0068	0.5523	0.7565	0.3
4	0.0155	0.0088	0.3363	0.0310	0.0064	0.5844	0.8941	0.5
7	0.0392	0.0208	0.3708	0.0460	0.0074	0.6887	0.8389	0.4
10	0.0252	0.0101	0.2889	0.0318	0.0064	0.7124	0.7942	0.2
12	0.0312	0.0145	0.3785	0.0459	0.0078	0.7480	0.8403	0.4

Table C-5: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.5)$ or a SNR = 19dB.

C-1-1 Performance on reference signal Input C_{var1}

Measurement Noise = $\eta(0, 01)$, 53dB

Disturbance	RMSE SDP-GD (m)			RMSE SLAM (m)			SDP-GD	SLAM
	X	Y	Depth	X	Y	Depth	R_d^2	R_d^2
0	0.00328	0.00041	0.01783	0.01618	0.00048	0.01729	0.99981	0.99983
1	0.00257	0.00041	0.01208	0.01617	0.00048	0.01249	0.99866	0.99862
2	0.00299	0.00041	0.01404	0.01610	0.00048	0.01195	0.99843	0.99882
3	0.00497	0.00041	0.01565	0.01636	0.00049	0.01047	0.99924	0.99963
4	0.00257	0.00041	0.02559	0.01618	0.00048	0.01121	0.99932	0.99987
6	0.00314	0.00041	0.02717	0.01587	0.00047	0.01176	0.99924	0.99988
7	0.00320	0.00072	0.04067	0.01382	0.00042	0.01573	0.99823	0.99981
8	0.00295	0.00072	0.04308	0.01382	0.00042	0.01538	0.99861	0.99990
10	0.00291	0.00041	0.03218	0.01636	0.00049	0.01219	0.99861	0.99972
12	0.00292	0.00072	0.04452	0.01405	0.00042	0.01737	0.99847	0.99978

Table C-6: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.01)$ or a SNR = 53dB.

Measurement Noise = $\eta(0, 0.5)$**,19dB**

Disturbance	RMSE SDP-GD (m)			RMSE SLAM (m)			SDP-GD	SLAM
	X	Y	Depth	X	Y	Depth	R_d^2	R_d^2
0	0.04567	0.01092	0.67191	0.16935	0.00574	1.07052	0.8570	0.6581
1	0.04706	0.01092	0.27157	0.16880	0.00571	0.59514	0.7096	0.2831
2	0.04585	0.01092	0.25657	0.16832	0.00569	0.48458	0.5512	0.2926
3	0.04760	0.01092	0.22344	0.17075	0.00579	0.49995	0.8582	0.5743
4	0.04860	0.01092	0.38566	0.16859	0.00569	0.54477	0.8191	0.6959
6	0.04471	0.01092	0.49312	0.16672	0.00563	0.62655	0.7689	0.6890
7	0.04574	0.01560	0.49828	0.14190	0.00468	0.74640	0.6128	0.5787
8	0.04244	0.01560	0.73294	0.14228	0.00471	0.78753	0.7194	0.7672
10	0.04389	0.01092	0.44157	0.17051	0.00579	0.70701	0.6998	0.6572
12	0.04789	0.01560	0.66964	0.14385	0.00477	0.66154	0.6586	0.7083

Table C-7: Average Root Mean Square Error (RMSE) in meters and squared Pearson correlation coefficient (R^2) of SDP-GD and Sonar SLAM output map performance, evaluated against the measurement data across varying levels of seabed distortion. The measurement used to create the output maps contains a noise of $\eta(0, 0.5)$ or a SNR = 19dB.

Bibliography

[1] S. Vakili, M. Khosravi, P. M. Esfahani, and M. Mazo, “Linear time-varying parameter estimation: Maximum a posteriori approach via semidefinite programming,” *IEEE Control Systems Letters*, vol. 8, pp. 73–78, 2023.

[2] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (slam) problem,” *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[3] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010. DOI: 10.1109/MITS.2010.939925.

[4] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual slam algorithms: A survey from 2010 to 2016,” *IPSJ transactions on computer vision and applications*, vol. 9, pp. 1–11, 2017.

[5] D. Nistér, “An efficient solution to the five-point relative pose problem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[6] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, “Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction,” in *2019 International conference on robotics and automation (ICRA)*, IEEE, 2019, pp. 5218–5223.

[7] C. Roman and H. Singh, “A self-consistent bathymetric mapping algorithm,” *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 23–50, 2007.

[8] J. Berwald, D. Stramski, C. D. Mobley, and D. A. Kiefer, “Influences of absorption and scattering on vertical changes in the average cosine of the underwater light field,” *Limnology and Oceanography*, vol. 40, no. 8, pp. 1347–1357, 1995.

[9] J. Y. Chiang and Y.-C. Chen, “Underwater image enhancement by wavelength compensation and dehazing,” *IEEE transactions on image processing*, vol. 21, no. 4, pp. 1756–1769, 2011.

[10] S. Zhang, S. Zhao, D. An, *et al.*, “Visual slam for underwater vehicles: A survey,” *Computer Science Review*, vol. 46, p. 100 510, 2022.

[11] RWMT Surveying Tools, *Side scan sonar for surveying*, Accessed: 26 March 2025, 2025. [Online]. Available: https://www.rwmt.se/surveying-tools/side-scan-sonar.

[12] LC Sonar, *Side scan sonar*, Accessed: 26 March 2025, 2025. [Online]. Available: https://www.lcsonar.com/PRODUCT-SideScanSonar/66.html.

[13] Blue Robotics, *Dvl-a50 doppler velocity log for auv and rov navigation*, https://bluerobotics.com/store/the-reef/dvl-a50, Accessed: 26 March 2025, 2025.

[14] Nortek, *New to subsea navigation? a guide to doppler velocity logs (dvl)*, https://www.nortekgroup.com/knowledge-center/wiki/new-to-subsea-navigation, Accessed: 26 March 2025, 2025.

[15] C. McQueen, *Orca4: Ros2 auv based on the bluerov2, ardusub and navigation2*, https://github.com/clydemcqueen/orca4, Accessed: 2024-06-26, 2023.

[16] D. R. Inc. and R. B. GmbH, *Desistek SAGA ROV model for UUV Simulator*, https://github.com/uuvsimulator/desistek_saga, Accessed: 2025-04-24, 2023.

[17] Blue Robotics, *Bluerov2*, Accessed: 2024-08-23, 2024. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/.

[18] HP Development Company, L.P., *HP ZBook Power 15.6 inch G9 Mobile Workstation PC specifications*, Accessed: April 4, 2025, 2025. [Online]. Available: https://support.hp.com/us-en/document/ish_5994641-5994705-16.

[19] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[20] M. M. M. Manhães, 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*, Ieee, 2016, pp. 1–8.

[21] M. Robotics, *Saga: Insansız sualtı aracı (rov)*, https://www.marenrobotics.com, Image of the SAGA ROV, a remotely operated vehicle capable of diving to 200 meters. Includes high-definition camera and real-time video transmission capabilities for underwater operations., 2023.

[22] G. Shanmugam, *Mass transport, gravity flows, and bottom currents: Downslope and alongslope processes and deposits*. Elsevier, 2020.

[23] T. I. Fossen, *Nonlinear modelling and control of underwater vehicles*. Universitetet i Trondheim (Norway), 1991.

[24] C.-J. Wu, "6-dof modelling and control of a remotely operated vehicle," Ph.D. dissertation, Flinders University, College of Science and Engineering., 2018.

[25] Wikipedia contributors, *Coefficient of determination — wikipedia, the free encyclopedia*, [Online; accessed 12-May-2025], May 2025. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination.

[26] S. Vakili, M. Mazo Jr, and P. M. Esfahani, "Optimal bayesian affine estimator and active learning for the wiener model," *arXiv preprint arXiv:2504.05490*, 2025.

[27] Gazebo Simulator, *Gazebo tutorials: Hydrodynamics*, Accessed: 2024-08-23, 2024. [Online]. Available: https://classic.gazebosim.org/tutorials?tut=hydrodynamics&cat=physics.

[28] Gazebo Simulator, *Theory of hydrodynamics*, Accessed: 2024-08-23, 2024. [Online]. Available: https://gazebosim.org/api/sim/8/theory_hydrodynamics.html.

[29] Gazebo Simulator, *Class gz::sim::systems::thruster*, Accessed: 2024-08-23, 2024. [Online]. Available: https://gazebosim.org/api/sim/8/classgz_1_1sim_1_1systems_1_1Thruster.html.

[30] ROS, *Rep 105: Ros package format*, Accessed: 2024-08-23, 2024. [Online]. Available: https://www.ros.org/reps/rep-0105.html.

[31] Blue Robotics, *Bluerov2 heavy retrofit kit*, Accessed: 2024-08-23, 2024. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit/.

[32] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[33] C. Cadena, L. Carlone, H. Carrillo, *et al.*, "Past present and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. DOI: 10.1109/TRO.2016.2624754.

[34] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

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

[36] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-dof slam with stereo-in-hand," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[37] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, Springer, 2000, pp. 298–372.

[38] Unique Group, *Teledyne blueview p900: 2d imaging sonar*, https://www.uniquegroup.com/product/teledyne-blueview-p900-2d-imaging-sonar/, Accessed: 26 March 2025.

[39] Analog Devices, *Adis16448: Compact, precision ten degree of freedom inertial sensor*, https://www.analog.com/en/products/adis16448.html, Accessed: 26 March 2025.

[40] M. von Benzon, F. F. Sørensen, E. Uth, J. Jouffroy, J. Liniger, and S. Pedersen, "An open-source benchmark simulator: Control of a bluerov2 underwater robot," *Journal of Marine Science and Engineering*, vol. 10, no. 12, p. 1898, 2022.