Experimental Evaluation of Underwater Semantic SLAM

by

Thomas Jeongho Song

B.A., Royal Military College of Canada, 2017

Submitted to the Department of Mechanical Engineering and the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE IN NAVAL ARCHITECTURE AND MARINE ENGINEERING

and

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by:	Thomas Jeongho Song Department of Mechanical Engineering August 9, 2024
Certified by:	John J. Leonard Professor of Mechanical and Ocean Engineering, Thesis Supervisor
Certified by:	Pulkit Agrawal Assoc. Professor of Electrical Engineering and Computer Science, Thesis Supervisor
Accepted by:	Nicolas Hadjiconstantinou Professor of Mechanical Engineering Graduate Officer, Department of Mechanical Engineering
Accepted by:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

Experimental Evaluation of Underwater Semantic SLAM

by

Thomas Jeongho Song

Submitted to the Department of Mechanical Engineering and the
Department of Electrical Engineering and Computer Science
on August 9, 2024 in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE IN NAVAL ARCHITECTURE AND MARINE ENGINEERING

and

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Autonomy is crucial for underwater vehicles due to the challenging and inaccessible nature of underwater environments. These environments pose significant difficulties for human-operated systems because of limited visibility, high pressure, and vast areas that are costly and risky to explore manually. Implementing autonomy in underwater vehicles presents unique challenges due to the marine environment's harsh and complex nature. Underwater communication is severely limited as water absorbs and scatters most electromagnetic signals used in terrestrial communications. This necessitates the use of acoustic communication, which has a lower bandwidth and is prone to delays and signal distortion. Similarly, GPS signals do not penetrate water, complicating navigation and creating dependence on inertial and sonar sensors, which suffer from noisy measurements that are guaranteed to drift over time. The unpredictable dynamics of underwater environments, including varying currents, lighting conditions and obstacles, further complicate autonomous navigation. As such, data collection while moving through a preplanned course is the traditional mission of the Autonomous Underwater Vehicle (AUV), defining the limitation of current technology. Higher-level missions such as search, surveillance, maintenance and manipulation require greater situational awareness, decision-making and navigation abilities, facilitated by processing semantic visual information and applying it to map generation and localization.

To address the limited autonomy of current AUVs and enhance their capability for complex missions, this thesis presents the development and evaluation of a real-time, monocular visual-inertial semantic Simultaneous Localization and Mapping (SLAM) system for underwater environments, implemented on the cost-effective BlueROV2 platform. The research aims to enhance AUV autonomy and enable complex underwater missions through improved navigation and semantic mapping capabilities. Key contributions include the integration of a custom-trained object detector for underwater environments, adaptation of a hybrid SLAM algorithm combining Gaussian and Non-Gaussian landmarks for underwater operation, preliminary assessment of the SLAM system's accuracy using motion capture-based ground truth measurements, and comparative evaluation of the developed semantic SLAM system against state-of-the-art alternatives in an indoor pool experiment using the BlueROV2. This

work addresses the challenges of underwater navigation and semantic mapping, offering a potential solution to extend the operational capabilities and mission complexity of affordable AUV platforms.

Thesis supervisor: John J. Leonard
Title: Professor of Mechanical and Ocean Engineering

Thesis supervisor: Pulkit Agrawal
Title: Assoc. Professor of Electrical Engineering and Computer Science

Acknowledgments

It was an immense privilege for me to have been chosen by MIT and the Royal Canadian Navy to enrol in this prestigious institution. The last two years can only be described as building my academic foundation, being enlightened by new fields of science and engineering, and being challenged to my limits both mentally and physically. My success and anything I have accomplished at MIT is certainly intertwined with the kind efforts of those who have pushed my back along the way, and I would like to recognize those who have been invaluable to my MIT experience here.

My research supervisor and head of MIT's Marine Robotics Group, professor John J. Leonard, has always been looking far ahead regarding my research and has always provided invaluable advice every time I needed it. He has always been a wise mentor, steering the direction of my research to be more productive, and gave me access to the right equipment and facilities and contact with the right people whenever I had inquiries. I would also like to deeply thank Associate Professor Pulkit Agrawal for the popular Sensorimotor Learning course in Spring 2023 as well as graciously accepting to be the reader of this thesis.

My friends at the Marine Robotics Group has always been welcoming and helpful to me. I must acknowledge: Kurran Singh, who has mentored me since day one of me joining the lab, allowed me to participate and learn from his experiments, and taught me how to operate the BlueROV2. QiangQiang "Chad" Huang had already graduated, but he was incredibly kind and patient with me as he helped me debug my modifications to his original GAPSLAM code. Jungseok Hong introduced me to my favorite restaurant in Boston, and as one of the post-docs, have been a great mentor in my research, studies and life in general. John Paul "JP" Morrison was my lab mate during the Visual Navigation for Autonomous Vehicles course, and with his exceptional programming skills, found impressive solutions to some of our lab problems. A proud member of the United States Navy submariners prior to being admitted to the MIT, he is a great conversationalist and storyteller.

Being involved in various club sports at MIT, I have been privileged to have gotten to know so many exceptional athletes and coaches. The Kickboxing Club has been my haven for athletics and extracurricular activities throughout my time at MIT. I would like to thank MIT research scientist Shan-Yuan "Hoho" Ho, who was a great kickboxing and Taekwondo coach that helped me take home a bronze medal at the National Collegiate Taekwondo Association tournament in 2023. Caio Silva and Tsegazeab Beteselassie, who are undergraduate students will always have a place in my heart as they acted as my right hand and left hand in successfully managing the club when it became time for me to step up and take on leadership. Additionally, the Brazilian Jiujitsu Club has been a very fun place to hang out, relieve stress and learn something new. I would like to thank Alex, Remeyn, Noah

and Max for teaching me grappling throughout this year.

To my United States Navy Academic Advisors Cdr Douglas Jonart and Cdr Christopher MacLean, I owe gratitude as they advised me on my academic course requirements for graduation, and approved my registration. I will not forget the remarkable presentations from the 2N events such as the Ship Design and Technology Symposium.

Although apart, my family has always been by my side virtually, sending messages of love and helping me with taking care of administration back in Canada. I would like to thank my mother Sanghee, brother Jaeson, father Kineung and sister Cathy, who have helped me get some peace of mind to completely concentrate on my studies here.

Thanks to you all as always,

TJ

Biographical Notes

Thomas Jeongho "TJ" Song has graduated from the Royal Military College of Canada with a Bachelor's Degree of Applied Science in Electrical Engineering in 2017. After five years of service as a Marine Systems Engineer on both Atlantic and Pacific coasts, he was sponsored by the Royal Canadian Navy to enrol in MIT and United States Navy's 2N program in 2022, to pursue a Masters of Science Degree in Naval Architecture and Marine Engineering.

During his time at MIT, he was admitted to the Marine Robotics Group and the Electrical Engineering and Computer Science Department in 2023 to work on visual SLAM and computer vision research for underwater vehicles. Additionally, he earned the Graduate Certificate in Technical Leadership from the MIT-Gordon Engineering Leadership Program.

Contents

List of Figures

List of Tables

List of Acronyms

AHRS Attitude and Heading Reference System
APE Absolute Pose Error
APS Acoustic Positioning Systems
AUV Autonomous Underwater Vehicles
BA Bundle Adjustment
BAD-SLAM Bundle Adjusted Direct SLAM
BoW Bag-of-Words
CLAHE Contrast Limited Adaptive Histogram Equalization
CIR Conditional Independence Relation
CNN Convolutional Neural Network
DA Data Association
DAF-SLAM Data-Association-Free SLAM
DVL Doppler Velocity Log
EKF Extended Kalman Filter
EVO Evaluation of Odometry and SLAM
FLANN Fast Library for Approximate Nearest Neighbors
FPS Frames per Second
GAPSLAM Gaussian Approximation and Particle Filters SLAM
GA Gaussian Approximation
GPS Global Positioning System
GTSAM Georgia Tech Smoothing And Mapping
GT Ground Truth
HAUD High-Accuracy Underwater Dataset for Visual-Inertial Odometry
iSAM2 Incremental Smoothing and Mapping version 2
IMU Inertial Measurement Unit
KNN K-Nearest Neighbors
LBL Long Baseline
LSD-SLAM Large Scale Direct SLAM
MAP Maximum a Priori
MC Monte Carlo
ML Machine Learning
MSCKF Multi-State Constraint Kalman Filter
NG Non-Gaussian
OAS-SLAM Opti-Acoustic Semantic SLAM
OKVIS Open Keyframe-based Visual-Inertial SLAM

ORB Oriented FAST and Rotated BRIEF
PA Patch Averaging
PID Proportional–Integral–Derivative
PR Precision-Recall
RANSAC Random Sample Consensus
R-CNN Region-Based Convolutional Neural Network
REMUS Remote Environment Monitoring Units
RMSE Root Mean Square Error
ROS Robot Operating System
ROV Remotely Operated Vehicle
RGB-D Red Green Blue and Depth
SIFT Scale-Invariant Feature Transform
SIM3 Similarity Transformation in 3D
SLAM Simultaneous Localization and Mapping
SONAR Sound Navigation and Ranging
SOTA state-of-the-art
SURF Speeded-Up Robust Features
SVIn2 Stereo Visual-Inertial Navigation System 2
UDCP Underwater Dark Channel Prior
UMAP Uniform Manifold Approximation and Projection
UKF Unscented Kalman Filter
USB Universal Serial Bus
UW-GAN Underwater Generative Adversarial Network
VINS Visual-Inertial Navigation System
VO Visual Odometry
WHOI Woods Hole Oceanographic Institution
YOLO You Only Look Once

Chapter 1

Introduction

1.1 Motivation

The demand for operating capabilities in undersea environments is on the rise [1], which is backed by growing environmental concerns [2], continued discoveries of deposits for undersea resources such as oil [3], minerals [4], and advancements in industries such as aquaculture [5], marine biotechnology [6] and ocean energy [7]. The unique challenges and opportunities of exploring these remote environments have captivated the attention of researchers and engineers alike.

The greatest asset for operating deep underwater is the Autonomous Underwater Vehicle (AUV), and their autonomy is advantageous due to several reasons. First, it allows AUVs to operate for extended periods without human intervention, enabling long-duration missions that are otherwise impractical. This autonomy leads to greater efficiency and cost-effectiveness by minimizing the need for constant human supervision, thereby reducing operational costs and freeing up human resources. Moreover, AUVs can access challenging and hazardous environments, such as deep-sea trenches and polar regions, where human presence is either dangerous or impossible.

When missions are properly planned, AUVs demonstrate reasonable reliability in performing tasks including environmental monitoring and data collection, which are essential for scientific research, resource exploration, and environmental protection. Onboard sensors and processing capabilities allow AUVs to make limited real-time decisions, allowing them to adapt to changing conditions, avoid obstacles, and optimize their paths to a certain extent without needing continuous communication with a control center; a significant advantage given the difficulties of underwater signal transmission. Additionally, autonomy increases operational flexibility, as AUVs can be deployed in swarms or fleets to cover larger areas or undertake more complex missions collaboratively. This systematic, efficient data collection enhances the quality and reliability of the information gathered, making AUVs indispensable tools for underwater exploration, research, and various industrial applications.

Consequently, marine autonomy has emerged as a strategic objective for organizations within the maritime industry in recent years. Lloyd's Register has categorized seven Levels of Autonomy [8] for surface marine vessels ranging from the most to least human involvement. Although it does not perfectly align, this framework is capable of being used to describe

levels of autonomy for undersea vehicles as well.

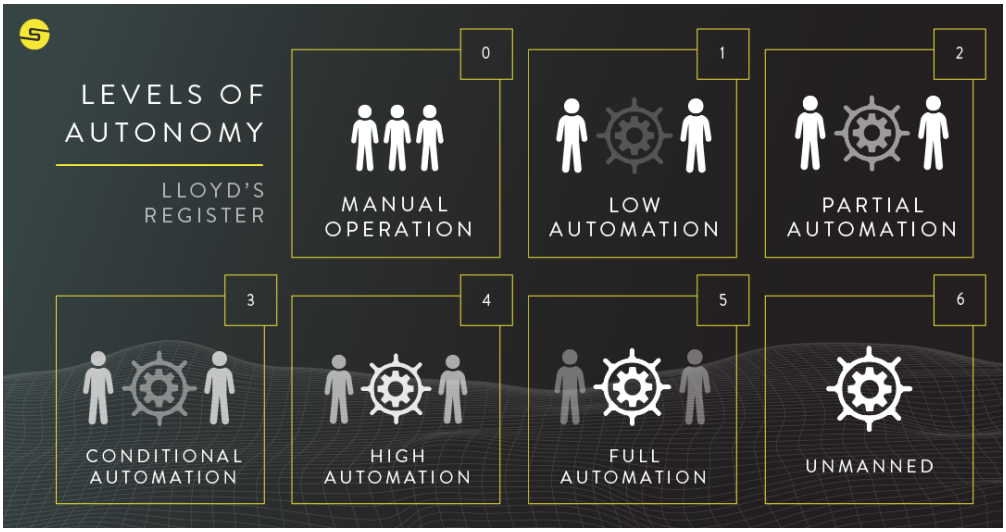


Figure 1.1: The seven levels of marine autonomy as described in the Lloyd's Register [8]. Level 0 is Manual Operation, without any automation. Level 1 is Low Automation, which include automated alarms and remote sensors. Level 2 is Partial Automation, also known as 'human-in-the-loop', which are automated tasks that leave room for operators to make a decision. Level 3 is Conditional Automation, also known as 'human-on-the-loop', which are automated tasks that do not require constant oversight but leaves room for human intervention when the system encounters unexpected situations. Level 4 is High Automation, where human intervention is limited within the vehicle's operational domain. Level 5 is Full Automation, where no personnel are required to be on board. Level 6 is unmanned and full automated.

Currently in the commercial realm, the autonomy level of AUVs has for the most part rested on Level 3, or Conditional Automation. Human involvement in operating AUVs is significant and encompasses a broad range of tasks. Operators are integral in meticulously planning and configuring the AUV's mission, setting parameters such as route, depth, and specific tasks. This phase includes thorough checks to ensure all systems, like sensors and batteries, are operational. Once the AUV is deployed, its operation shifts to a more autonomous mode, yet human oversight continues from a support ship or shore-based station. Operators monitor the AUV's performance via limited acoustic communication to track its adherence to the mission plan and manually respond to anomalies as well as plan and execute recovery after completion of the mission. After recovery, depending on the mission the researchers may need to review hours of video and Sound Navigation and Ranging (SONAR) data, as well as logs from other mission-specific sensors.

Human operators continue to play such a significant role in AUV operation because in comparison to an orderly laboratory setting where access is readily available and sensors operate relatively noise free, the underwater environment is the exact opposite. Navigating and operating autonomously underwater is challenging due to the inherent degradation of visual data recorded by underwater cameras and the accumulation of noise from other onboard sensors. An object that is easily identified by camera a given distance away in a

terrestrial environment may become unidentifiable at the bottom of the seafloor. Underwater visibility is often limited due to factors such as turbidity, suspended particles known as marine snow, and the absorption and scattering of light [9]. In optically clear waters, the maximum range of visibility is only 80 meters [10], making it difficult for sensors, primarily cameras, to capture clear and detailed images from a distance, which impacts the quality of semantic information. Natural light decreases with depth, and only a narrow band of visible light wavelengths penetrate meaningfully past 20 meters, making everything appear green and blue tinted underwater [9]. Past 200 meters, very little light penetrates regardless of the wavelength, making depth immeasurable and making all features disappear into a monochromatic region. As such, visual sensors are near ineffective in open waters for mapping, and AUVs must be near the seafloor in order to have the opportunity to capture visual landmarks that can bring situational awareness and indicate the robot's pose. In optically clear waters with good visibility, both the overhead perspective of bird's-eye view and the forward-facing seafloor level perspective may yield sufficient useful features. However, in darker conditions artificial illumination is required and only the bird's-eye view that typically has less variability of depth and better return of reflected light may be useful. Artificial lighting is ineffective unless the water is very clear and results in variable and often low illumination conditions in brackish waters for any perspective, which challenges cameras in capturing high-quality images for semantic analysis.

Additionally, light behaves differently underwater compared to in the air, causing distortions and refraction that can affect the accuracy of image data. Accurate and distortion-free visual data is crucial for semantic understanding of what has been observed, but underwater conditions often blur perspectives and lessen contours and colors, necessitating techniques such as enhancement and restoration that leverage heuristics and machine learning, which have resulted in mixed levels of success [11] varying significantly from dataset to dataset. Color change increases away from the air-water surface boundary, while refraction and distortion increases towards the surface boundary.

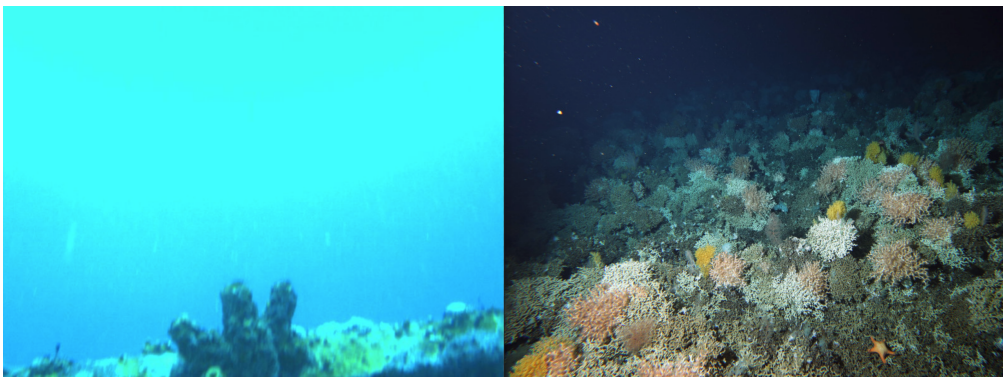


Figure 1.2: Left: forward facing perspective in a bright environment [12]. Right: downwards perspective in a dark environment [13]. Both have significant regions that lack features due to illumination effects. The bright environment is also closer to the surface, so air bubbles and their reflections can be seen. The dark environment is filled with drifting marine snow.

Underwater scenes shown in figure 1.2 typically lack the distinct visual features, contrasts, and textures found on land, complicating computer vision algorithms' ability to identify and

track objects or features, particularly for semantic segmentation tasks [14]. Traditional positioning methods such as Global Positioning System (GPS) and radio communication are generally unavailable underwater due to the rapid attenuation of electromagnetic signals [15]. Seawater-air and seawater-seabed interfaces are more permissive, but still limit signal transfer to several meters [16]. Communication buoys that leverage a combination of GPS and acoustic modules have been successful in positioning underwater vehicles, but the inherent limitations of underwater acoustics relative to the distances required for exploration missions make their widespread usage infeasible [17]. Acoustic communication is generally hindered by insufficient bandwidth and range, requiring the entire navigation, mapping, and mission planning process to be performed within the robot's limited onboard capabilities; emphasizing the need for fast and efficient algorithms [11]. A tethered Remotely Operated Vehicle (ROV) offers more reliable communication with the user and access to additional computational resources, but it reduces the vehicle's autonomy as the tether's position, unknown to the robot, must be accounted for to avoid obstruction and entanglement. Dead reckoning and maintaining the true pose of the underwater robot is challenging over longer periods, even with advanced and expensive sensors, due to drift from accumulating errors. Acoustic Positioning Systems (APS), such as Long Baseline (LBL) [18], can correct ground-truth errors but require prior installation and offer low resolution, limiting their application in underwater localization [19]. Additionally, dynamic marine life such as seaweed, coral, or fish can occlude the view and interfere with sensor readings, adding complexity to detecting and segmenting objects accurately. Compared to terrestrial environments, there is a scarcity of labeled training data for underwater scenes, necessitating extensive data collection efforts for training Machine Learning (ML) models for semantic segmentation in underwater conditions [20].

Despite the challenging conditions, AUVs such as the REMUS [22] made by the WHOI shown in figure 1.3 have a proven track record of successful missions. The typical mission of an AUV encompasses a variety of critical tasks, primarily focused on mapping and surveying, environmental monitoring, inspection and maintenance, scientific research, defense and security, and resource exploration. In mapping and surveying, AUVs create detailed maps of the seafloor using bathymetric techniques and side-scan Sound Navigation and Ranging (SONAR) surveys, which capture high-resolution images of underwater features. Environmental monitoring involves collecting data on water quality parameters like temperature, salinity, and chemical composition, as well as studying marine habitats such as coral reefs and fish populations. For inspection and maintenance, AUVs examine underwater infrastructure, including pipelines, cables, and ship hulls, ensuring their integrity and identifying areas needing maintenance. Scientific research missions utilize AUVs to gather oceanographic data, conduct geological and geophysical surveys, and support studies on ocean currents, marine biology, and climate change. In defense and security, AUVs play a crucial role in mine countermeasures, detecting and neutralizing underwater mines, and conducting surveillance and reconnaissance operations to monitor strategic areas. Lastly, resource exploration missions employ AUVs to survey potential underwater drilling sites for oil and gas, as well as search for underwater mineral deposits.

Although already capable of providing great utility, there is potential for AUVs to achieve levels beyond Conditional Automation without requiring state-of-the-art (SOTA) hardware. For robots to rise into higher autonomy levels they must at minimum operate robustly in

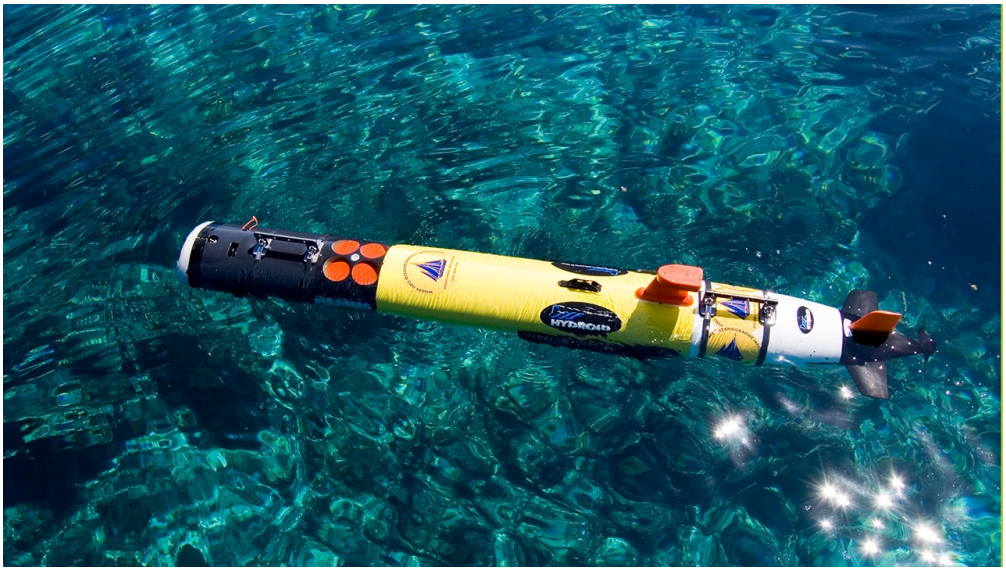


Figure 1.3: Image of a REMUS 100 unit created by WHOI and Kongsberg Maritime. During Operation Iraqi Freedom in 2003, the U.S. Navy used REMUS vehicles to detect mines in the Persian Gulf harbor. Navy officers said they preferred REMUS AUVs because each could do the work of 12 to 16 human divers, and they were "undeterred by cold temperatures, murky water, sharks, or hunger." [21]

real-time, prioritize safety, recognize and avoid danger, be capable of independently planning and executing steps required to achieve mission objectives, navigate to and from any known coordinates while maintaining a situational awareness of its surroundings, and semantically differentiate regions and objects of importance observed in their visual or SONAR field of view. With the advent of highly affordable, portable and powerful processors and recent advancements in image enhancement algorithms [23], image semantic segmentation [24], object detection [25], and SLAM techniques [26], some of the foundational tools for achieving the requirements that support next level autonomy have been made available.

This research is motivated by the prospect of leveraging these new foundational tools to significantly elevate AUV autonomy, increasing safety, convenience and capability without sacrificing cost-effectiveness. To meet the challenges inherent in underwater environments, this thesis proposes the design and evaluation of a visual semantic SLAM system for underwater robots as a solution.

1.2 Summary of Contributions

This thesis describes the development and evaluation of a real-time semantic SLAM system for operating in an underwater environment, sparsely populated by known objects on a real robot. Our main contributions are:

1. This work develops the design of a visual semantic SLAM system capable of running on underwater robots such as AUVs and ROVs.

2. This work explores ground truth trajectory evaluations of visual SLAM systems in order to validate their effectiveness via trajectory error comparison.

3. This work establishes benchmarks for the proposed SLAM system by comparing it with other SOTA underwater SLAM systems, aiming to identify areas for further improvement.

1.3 Thesis Overview

The structure of the thesis is as follow. Chapter 2 introduces previous work that serves as inspiration for this thesis: the visual SLAM problem, object detectors, related work in underwater SLAM, and the GAPSLAM method.

Chapter 3 focuses on the preliminary study for evaluating GAPSLAM against ORB-SLAM3 and motion capture ground truth.

Chapter 4 describes the design of an underwater visual semantic SLAM system for BlueROV2 that leverages GAPSLAM and object detectors, covering system architecture, and the interfacing hardware and software.

Chapter 5 compares underwater GAPSLAM to the SOTA underwater visual semantic SLAM method in an indoor pool experiment, via conducting an analysis of the results

Chapter 6 discusses the points of improvement for our underwater visual semantic SLAM system.

Chapter 7 summarizes our contributions and findings, as well as presenting directions for future research.

Chapter 2

Literature Review

2.1 Visual SLAM

Visual SLAM enhances the autonomy of robots via mapping and localization. Localization against previously seen parts of the map is essential in correcting the trajectory drift guaranteed on AUV, while maps record the observations made by the AUV. This is inherently useful because it is the objective for mapping missions, or it can be a useful tool for a human operator or the robot's on board systems in different mission types to quickly replan a new course by making use of the map. There is a significant collection of literature on visual SLAM, dating back to seminal work by Davison [27] and others. Recent surveys include [28] and [29]. Notable work in underwater visual SLAM includes [30] and [31].

2.1.1 Factor Graphs

Factor graphs are graphical models that are well suited to modelling complex estimation problems, such as the SLAM problem that deals with robot poses over time relative to landmarks positions in the environment [32]. A factor graph is a bipartite graph consisting of factors interfacing with one or more variables as described in figure 2.2. The variables are the nodes of the graph that represent the unknown random variables in the estimation problem, whereas the factors represent probabilistic information on those variables, derived from measurements or prior knowledge [33]. In the case of the SLAM problem, the variables nodes are the robot poses and landmarks, while the factor edges are the probabilistic observations from the robot to the landmarks, odometry measurements estimated on board as the robot moves from point A to B, and loop closures when the robot feature matches and recognizes that it has returned to a location that it has been in previously. Consequently factor graph SLAM problem in simple form can be expressed as the following joint probability distribution:

$$p(\mathbf{X}, \mathbf{L}|\mathbf{Z}, \mathbf{U}) \propto \prod_i p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \prod jp(\mathbf{z}_j|\mathbf{x}_j, \mathbf{l}_j) \tag{2.1}$$

where $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_T$ is the set of robot poses, $\mathbf{L} = \mathbf{l}_1, \ldots, \mathbf{l}_N$ is the set of landmarks, $\mathbf{Z} = \mathbf{z}_1, \ldots, \mathbf{z}_M$ is the set of observations, $\mathbf{U} = \mathbf{u}_1, \ldots, \mathbf{u}_T$ is the set of odometry measurements, $p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)$ is the odometry model, and $p(\mathbf{z}_j|\mathbf{x}_j, \mathbf{l}_j)$ is the observation model. The factor

graph representation translates the problem into one that can be resolved through graph optimization. The optimization objective can be written as:

$$\mathbf{X}, \mathbf{L} = \arg\max_{\mathbf{X},\mathbf{L}} \prod_i \psi_i(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i) \prod j\phi_j(\mathbf{z}_j, \mathbf{x}_j, \mathbf{l}_j) \tag{2.2}$$

where $\psi_i(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i)$ are the odometry factors and $\phi_j(\mathbf{z}_j, \mathbf{x}_j, \mathbf{l}_j)$ are the observation factors. The optimization typically involves minimizing a nonlinear least-squares problem, which can be written as:

$$\mathbf{X}, \mathbf{L} = \arg\min_{\mathbf{X},\mathbf{L}} \sum_i |\mathbf{e}_i(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i)|^2 + \sum j|\mathbf{e}_j(\mathbf{z}_j, \mathbf{x}_j, \mathbf{l}_j)|^2 \tag{2.3}$$

where $\mathbf{e}_i(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i)$ are the error terms for the odometry model and $\mathbf{e}_j(\mathbf{z}_j, \mathbf{x}_j, \mathbf{l}_j)$ are the error terms for the observation model. The factor graph model describes the mathematics of the SLAM problem, but SLAM is a robotics problem that is comprised of additional processes. SLAM is described as having a front-end and a back-end, which are the two major processes required to implement SLAM on a given platform. For both the front-end and back-end there are various approaches each with their advantages and disadvantages.

2.1.2 Front-End

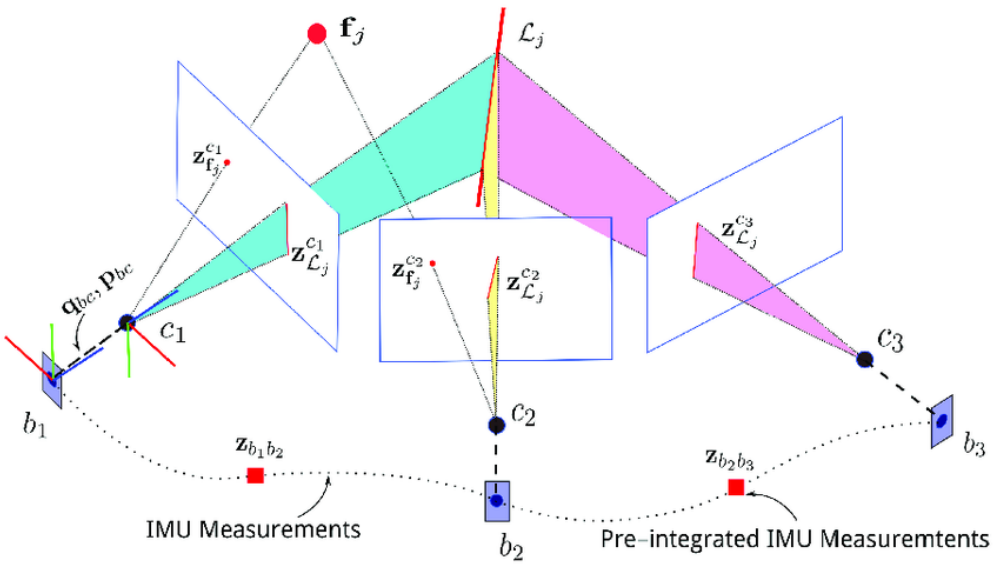


Figure 2.1: Visualization of VO showing IMU integration [34], where f is the focal point, c is the center of the camera, z_{f_j} represents odometry factors from matched features and $z_{\mathcal{L}}$ are observation factors generated from features or landmarks in the image frame.

The front end is responsible for generating the components required for the SLAM equations. Features in the current frame are matched with features from a previous frame, assuming that only the observer has moved and that most of the environment is stationary. The observer's movements are estimated by the matched features and camera focal point

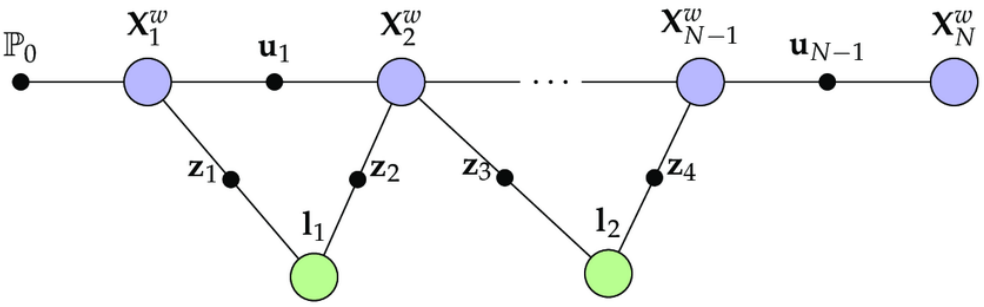


Figure 2.2: A simple visualization of the factor graph as it is used in SLAM, given by Lai [26]. Robot state variables X are depicted as purple nodes, and landmark variables L are depicted as green nodes. Odometry factors u and observation factors z and depicted as black vertices.

and center, which connect the current robot state variable to the previous one as shown in figure 2.2. Observation factors on the other hand, are generated by mapping the current pose of the camera to landmark positions in the frame and do not require frame-to-frame matches. Observation factors connect robot state variables to landmarks.

Feature Extraction

Data acquisition and Feature Extraction are critical components in visual SLAM. Features from image frames can be used to form both observations $\mathbf{z}_j$ and odometry $\mathbf{u}_i$. Effective Feature Extraction ensures accurate and reliable SLAM performance by identifying key points in images that can be consistently tracked across multiple frames.

Scale-Invariant Feature Transform (SIFT) [35] is a widely used technique for detecting and describing local features in images. SIFT is robust to scaling, rotation, and changes in illumination, making it highly effective for matching features across different views. This method identifies key points in an image and computes descriptors based on the local gradient information around each key point. The robustness and accuracy of SIFT have made it a standard choice for various computer vision applications, including object recognition and image stitching.

SURF (Speeded-Up Robust Features) [36] is a faster alternative to SIFT, designed to provide similar robustness to scaling, rotation, and noise while being computationally more efficient. Speeded-Up Robust Features (SURF) uses integral images to speed up the computation of image convolutions and relies on the Hessian matrix for key point detection. The descriptors are based on the sum of Haar wavelet responses within a neighborhood around each key point. SURF's efficiency and robustness make it suitable for real-time applications where computational resources are limited.

Oriented FAST and Rotated BRIEF (ORB) [37] is another popular Feature Extraction method known for its efficiency and speed, making it suitable for real-time applications. ORB combines the FAST key point detector and the BRIEF descriptor, with modifications to ensure rotational invariance. It is particularly effective for applications where computational efficiency is crucial, such as mobile robotics.

Observation Factors

From the extracted features, associations can be made between the robot pose and landmarks, which is essential for forming the observation model $p(\mathbf{z}_j|\mathbf{x}_j, \mathbf{l}_j)$ and consequently creating the observation factors $\phi_j(\mathbf{z}_j, \mathbf{x}_j, \mathbf{l}_j)$. Observation factors adds constraints into the factor graph, which is essential during the back end's optimization. They are most effective when same landmarks are observed from multiple robot poses

Feature Matching

Feature Matching is the process of pairing extracted features from one frame to their equivalents in the following frame. This pairing allows estimation of translation and rotation of the camera. The choice of Feature Matching method can significantly impact the robustness and efficiency of the SLAM system, especially in dynamic and complex environments.

The Bag-of-Words (BoW) model is a powerful technique for Feature Matching and place recognition. Extracted features are clustered into visual words using a clustering algorithm such as K-Means, creating a visual vocabulary. Each image is represented as a histogram that counts the occurrences of these visual words. For Feature Matching, the BoW histograms of different images are compared using similarity measures like Euclidean distance or cosine similarity, enabling the system to identify similar images or regions. This is particularly useful for place recognition and loop closure in SLAM, as it allows the system to recognize previously visited locations based on visual similarity. By detecting loop closures, the SLAM system can correct accumulated drift, refine the map, and improve the accuracy of the estimated trajectory.

Another method is K-Nearest Neighbors (KNN) [38], which matches features between consecutive frames by finding the nearest neighbors in the feature space. This technique, detailed in Altman's introduction to KNN, is straightforward and effective for real-time applications. KNN works by comparing each feature in the current frame to a set of features in the previous frame and selecting the closest matches based on a distance metric, such as Euclidean distance. This method is widely used due to its simplicity and effectiveness, although it may become computationally expensive as the number of features increases.

For large datasets, the Fast Library for Approximate Nearest Neighbors (FLANN) [39] is an efficient choice. FLANN is used for approximate nearest neighbor searches, offering a balance between speed and accuracy, as detailed by Muja and Lowe. FLANN uses a combination of randomized algorithms and hierarchical data structures to quickly find approximate nearest neighbors, significantly reducing the search time compared to exact methods. This makes FLANN particularly suitable for real-time SLAM applications where fast processing of large numbers of features is required.

Lastly, Random Sample Consensus (RANSAC) [40] is a robust method for estimating the parameters of a model, which is particularly effective in filtering out outliers in feature matches. The seminal work by Fischler and Bolles provides a comprehensive overview of RANSAC. This method iteratively selects random subsets of data points and fits a model to these points, checking the fit against the entire dataset to identify inliers and outliers. RANSAC's ability to handle high levels of noise and outliers makes it invaluable for SLAM, where data can often be noisy and incomplete.

Visual Odometry

The front end models the robot's motion $p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)$, providing the odometry factors $\psi_i(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{u}_i)$. Visual odometry (VO) [41] is a technique that uses Feature Extraction and Feature Matching between frames to estimate these motion factors. Visual odometry ingeniously leverages the fact that most cases, a majority of the features detected in one frame will be found again in the next frame. Visual odometry will fail if there are insufficient common features between frames. By matching each common feature between frames, it is possible to estimate the translation and rotation of the robot moving from one frame to the next. However, sensor noise, and outliers in the form of false positive and false negative matches must be managed as they could cause the trajectory to drift due to error accumulation. These techniques will be discussed in the data association and back-end sections below.

Monocular VO [42] uses a single camera to estimate motion. The main challenge with this method is the unknown scale of the translation between frames, which can lead to inaccuracies in motion estimation if not properly handled. Techniques to mitigate this issue include using additional information from other sensors or imposing constraints based on known scene geometry.

Stereo VO [43] employs a stereo camera setup to overcome the scale ambiguity inherent in monocular systems. The known baseline between the cameras allows for accurate depth estimation and motion estimation with scale. This setup enables the system to triangulate feature points and compute the 3D structure of the scene, resulting in more robust and accurate motion estimation.

Red Green Blue and Depth (RGB-D) VO [44] uses an RGB-D camera that provides both color and depth information. This additional depth information enables more robust and accurate motion estimation compared to monocular and stereo systems, particularly in environments with low texture or poor lighting conditions. RGB-D VO is well-suited for indoor environments where depth data can significantly enhance Feature Matching and pose estimation.

Sensors that measure motion, such as Doppler Velocity Log (DVL) and IMUs [45], can derive odometry by themselves or be integrated with VO. These sensors can also be fused with other sensor outputs via EKFs [46] and Unscented Kalman Filter (UKF)s [47] to improve odometry performance. The integration of these sensors helps in providing more accurate and reliable motion estimation, particularly in challenging environments where VO alone may not suffice.

Data Association

In the front-end of visual SLAM, data association [48] leverages Feature Extraction and Feature Matching to play a crucial role in linking observations from the camera with existing elements in the map or previously observed features. Data association begins with Feature Extraction from incoming camera frames using algorithms such as SIFT or ORB. Each extracted feature is then described using a descriptor, which encodes its local appearance. These descriptors are compared with those from previous frames or the map to find correspondences. Techniques such as RANSAC are employed to match the extracted features

accurately, filtering out outliers and ensuring that only reliable matches are considered. As features are matched, the system updates the map by adding new landmarks for unmatched features and refining existing landmarks for matched ones. This is known as loop closure, which adds helpful constraints to narrow down uncertainty for the back-end processing.

2.1.3 Back-End

The back end in SLAM is responsible for constructing, maintaining, and optimizing the factor graph created by the front end. This graph consists of nodes representing robot poses ($\mathbf{X}$) and landmarks ($\mathbf{L}$), with edges representing factors derived from motion and observations. The optimization process aims to maximize the joint probability distribution or minimize the error terms in the nonlinear least-squares problem, as represented in equations (2.2) and (2.3). This involves finding the most likely configuration of variables that fit the factors, ensuring estimated poses and landmarks are consistent with observed data. Common techniques include Bundle Adjustment (BA) [49], which jointly optimizes the entire trajectory and map by minimizing reprojection error of all observations. Iterative methods like Gauss-Newton [50] or Levenberg-Marquardt [51] are often used to refine estimates. Specific algorithms such as Georgia Tech Smoothing And Mapping (GTSAM) [52] operate as a batch optimization framework, while Incremental Smoothing and Mapping version 2 (iSAM2) [53] provides an incremental approach suitable for real-time applications.

Error minimization in the back end focuses on reducing the sum of the error terms $\|\mathbf{e}_i\|$ and $\|\mathbf{e}_j\|$, which represent the differences between the predicted and observed values. This process incorporates noise and uncertainties into the model, ensuring that the system can robustly handle real-world variations and inaccuracies. By effectively minimizing these errors, the SLAM system can achieve more accurate and reliable estimates of the robot's trajectory and the map of the environment.

After the optimization process, the back end updates the estimated map and the robot's trajectory. This step provides an accurate and updated representation of the environment and the robot's path, reflecting the most recent observations and corrections. The continuous update of the map and trajectory is essential for maintaining an accurate understanding of the environment, especially in dynamic or previously unexplored areas.

Loop closure detection and correction are integral parts of the SLAM back-end, where it incorporates loop closure detections described in the front-end into the factor graph. Loop closures add additional constraints that help correct for drift and improve the overall accuracy of the system. By recognizing previously observed landmarks in new frames that are associated with a specific location on the map, the SLAM system can refine its estimates and reduce cumulative errors, leading to a more consistent and accurate map.

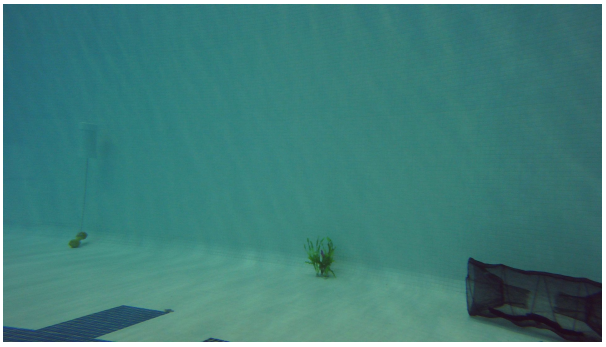
2.1.4 Interaction Between Front-End and Back-End

To summarize, the front end processes the raw sensor data, extracts features, performs Feature Matching, and generates factors that describe the constraints between the variables. These factors are then fed into the back end, which constructs the factor graph and performs optimization to solve for the best estimates of the variables. The front end continuously

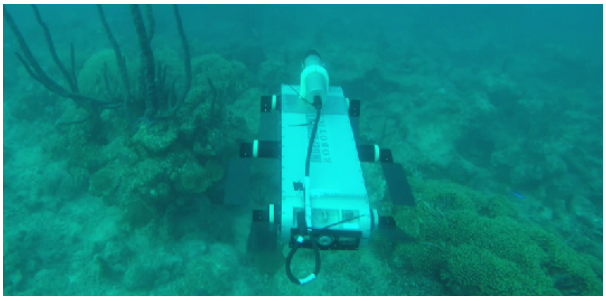
generates the factors as new data comes in, and the back end continuously optimizes the factor graph to refine the map and trajectory estimates.

2.2 Related Work

Underwater visual SLAM has been less studied than terrestrial visual SLAM, however there is some notable prior work, which includes the following references [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] that are further explored below. There are two categories of testing: laboratory settings, which are more common and take place in indoor pools and tanks such as the one shown in figure 2.3a, and real-world AUV scenarios, which are more scarce and come from video collected by AUVs operating in various mission scenarios such as the one shown in figure 2.3b.



(a) Image of the underwater environment at the MIT Zesiger Center Pool containing a lobster trap (right), seaweed (middle) and buoy (left) [63]. The white buoy is difficult to observe in this environment.



(b) Image of an autonomous robot in an outdoors underwater lake environment [12].

Figure 2.3: Comparison of underwater environments

2.2.1 Underwater Monocular SLAM

A recent experiment by Grimaldi et al. [54] evaluated six distinct image enhancements on both real-world AUV and lab datasets of varying illumination: artificial lighting, homogeneous sunlight and mixed lighting. In order to counteract the degradation of Feature Extraction that results in underwater environment due to visibility and illumination limitation, the following image enhancement algorithms were tested: the heuristic Underwater Dark Channel Prior (UDCP) algorithm [64], the heuristic Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm [65], the Underwater Generative Adversarial Network (UW-GAN) algorithm that was trained on three different types of environments using machine learning, and lastly the median filter from Köser et al [66]. Grimaldi et al. [54] also kept the unenhanced images in the experiment as a baseline. The enhanced and baseline images were used to test multiple SLAM systems. A successful run count was not granted unless the entire sequence was completed, and failure consisted of three categories: could

not initialize or operated from incorrect initialization, lost tracking midway, and complete failure.

The SLAM methods chosen for the test were monocular, and consisted of: ORB-SLAM2 [67], ORB-SLAM3 [68], Large Scale Direct SLAM (LSD-SLAM) [69], Bundle Adjusted Direct SLAM (BAD-SLAM) [70], and GradSLAM [71]. ORB-SLAM2 is purely a feature based method that uses ORB descriptors, while ORB-SLAM3 incorporate additional semantic information to improve the robustness and accuracy of the system. LSD-SLAM estimates camera motion and 3D structure directly from the intensity values of the image pixels without relying on feature detection and mapping. This method is known for its ability to handle large-scale environments and low-texture scenes typically expected in underwater scenes. BAD-SLAM is another direct method that specializes in leveraging the RGB-D inputs and performing BA. GradSLAM leverages computational graphs and deep learning to apply differentiable VO and differentiable optimization to the SLAM problem for joint and multi-objective optimization of all parameters including camera motion, depth estimation and mapping. Direct SLAM systems require depth maps, which were generated by UW-Net [72] and UDepth [73].

In contrast to their good performance in typical featureless environments, the results of the experiment by Grimaldi et al. [54] showed that Direct SLAM methods such as LSD-SLAM, BADSLAM and GradSLAM all failed because the depth estimators malfunctioned as they were not designed for top-down views. Additionally, Grimaldi et al. [54] recommended that users avoid direct SLAM algorithms for underwater use because they were too sensitive to the adverse lighting conditions, even with the image enhancement [54].

Other problems included insufficient overlap between frames, which hindered the algorithms' ability to establish robust correspondences of features. This is due to the low Frames per Second (FPS) video recorded due to the limitations of on board hardware combined with jerky rotational motion of the AUV used in the experiment.

Lighting conditions were critical for successful underwater SLAM performance, and performing an optimal initialization procedure was key to success for the ORB-SLAM methods. The only successful runs without failure, or losing tracking were ORB-SLAM methods in the laboratory tank with optimal and homogeneous sunlight [54]. Even these successes were marred by a relatively large absolute error range of 1.34m to 1.61m after being aligned and scaled using the Similarity Transformation in 3D (SIM3) Umeyama technique [74] [75].

2.2.2 Underwater Visual SLAM with Sensor Fusion

According to the types of sensors, underwater SLAM can be divided into Sound Navigation and Ranging (SONAR) SLAM [20], visual SLAM [76], and SLAM fusing multiple sensors. SONAR uses a transmitter to emit sound waves and a receiver to receive echo signals, then analyzes and processes the echo signals to describe the contour and structure of the target. Sound wave propagation under water is unaffected by light, making SONAR is a good choice for underwater SLAM. Although relatively low-resolution and high-cost, SONAR is a good candidate for sensor fusion with visual SLAM along with inertial sensors for odometry. Combinations of sensors in significant sensor fusion experiments range from the simple monocular visual-inertial, to complex systems leveraging stereo cameras, SONAR and EKF odometry from IMU, DVL and pressure sensor.

High-Accuracy Underwater Dataset for Visual-Inertial Odometry (HAUD) for visual-inertial odometry by Song et al. [55] was the first experiment to use the Vicon motion capture system underwater for accurate ground truth, although the test environment was an inflatable pool that was relatively small. Ten different sequences consisting of trajectories of different shapes were planned for the testing of ORB-SLAM2 versus Visual-Inertial Navigation System (VINS) [77] using a stereo ZED camera [78] with an on board IMU. This experiment demonstrated that there was a lot of variance between the sequences. A few sequences favored ORB-SLAM2, while the majority favored VINS. However, ORB-SLAM2 was more robust than VINS as it did not fail any sequence by losing tracking, while VINS failed two of the ten. The most significant results of this experiment was the millimetre resolution of the experiment results to the ground truth.

Stereo Visual-Inertial Navigation System 2 (SVIn2), developed by Rahman et al. [56], is a SLAM system designed for challenging underwater environments such as wrecks and caves. It integrates acoustic, visual, inertial, and depth sensors, and employs CLAHE image enhancement techniques similar to Grimaldi et al. [54]. Based on Open Keyframe-based Visual-Inertial SLAM (OKVIS) [79], SVIn2 features a robust initialization system using a stereo camera, downwards SONAR, IMU, and depth sensor, activating only when sufficient visual features are present. In comparisons with OKVIS, monocular VINS [80], and Multi-State Constraint Kalman Filter (MSCKF) [81] on the EuRoC [82] datasets without SONAR and depth sensor integration, SVIn2 showed lower Root Mean Square Error (RMSE) [83] than OKVIS and MSCKF, but did not consistently outperform monocular VINS. However, SVIn2 demonstrated exceptional performance in real-world outdoor lake and cavern datasets captured with the Aqua2 vehicle [84], consistently recognizing loop closures and maintaining trajectory consistency with the pseudo GT. In contrast, monocular VINS struggled with scale issues and exposure variations, while OKVIS experienced scale drift that prevented accurate loop closure detection, resulting in trajectories that significantly deviated from the pseudo-GT.

2.2.3 Underwater Semantic Visual SLAM with Sensor Fusion

While terrestrial semantic visual SLAM has been rigorously explored in works such as Civera et al [85], far fewer works exist to challenge the same problem underwater. Most of the existing underwater SLAM results [57] [58] [59] [60] have used pre-existing CAD models and SONAR for object detection.

Some real world tests of semantic SLAM have been applied on underwater pipe tracking. In a 2021 study by Vallicrosa et al. [61], researchers presented a method combining feature-based SLAM with 3D object recognition. This technique allowed an AUV to autonomously navigate and map submerged pipeline structures, using sensors such as DVL, and attitude and heading Attitude and Heading Reference System (AHRS) to provide real-time, drift-less navigation and consistent mapping.

Additionally, the SubPipe [62] dataset, introduced in 2024, provided a valuable resource for developing and testing semantic SLAM algorithms tailored for submarine pipeline inspection. This dataset includes high-resolution camera footage and SONAR images annotated for pipeline segmentation, facilitating the training of machine learning models for accurate

pipeline detection and tracking.

At present, the OAS-SLAM method by Singh et al. [63] represents a recent SOTA semantic visual SLAM method as it encompasses most of the underwater SLAM technologies explored by previous research: it fuses complimentary visual and acoustic data for observation factors, combines IMU,DVL and pressure sensor via EKF for odometry factors, uses iSAM2 for back-end optimization, is capable of semantically recognizing objects of various classes without prior knowledge and can output a map that has object landmarks overlaying the robot trajectory.

In the front end, MaskCut [86] is used to segment an image frame and generate masks of the segmentation. PA and Uniform Manifold Approximation and UMAP are two methods for accommodating masks of varying sizes while ensuring the output features maintain a constant dimension to facilitate data fusion with the SONAR. PA and UMAP are tested against YOLOv8 for precision and recall across four indoor pool tests and one outdoor tank test, where it is discovered that PA outperforms UMAP. YOLO outperformed PA in precision, but PA had more datasets where its recall was superior.

2.3 GAPSLAM

GAPSLAM, is a visual semantic SLAM algorithm developed in an office setting by Huang et al [87]. A single camera without a depth sensor is sufficient to run GAPSLAM; the bearing-only requirement simplifies the computational and hardware constraints. GAPSLAM holds research value in the field of underwater SLAM because of its novelty, efficiency and potential to be operable on lower-cost robots that are not fitted with SOTA sensors. As such, it has been selected for this experimental implementation of underwater semantic SLAM.

GAPSLAM, visualized in figure 2.4 blends the advantages of the Gaussian Approximation (GA) method on scalability for tackling high-dimensional posteriors, and Non-Gaussian (NG) particle filter method on inferring marginal posteriors encountered in SLAM. These two methods operate in complement to each other, and support each other in achieving optimization. Specifically, GA provides robot pose distributions on which particle filters are conditioned to sample landmark marginals. In return, if a sample attains a higher probability than the Maximum a Priori (MAP) estimate when evaluating the posterior, the MAP point among these samples can be used to reset linearization points in the non-linear solver of the GA, facilitating the pursuit of global optima. The following subsections describe equations derived by Huang [87], which review GA, marginal posteriors of NG landmarks and reinitialization.

2.3.1 Gaussian Approximation

The basis of GAPSLAM is landmark-based SLAM. Let $\mathbf{X}_t := (X_0, X_1, \ldots, X_t)$ be robot pose variables up to time step t where $X_i \in \mathrm{SE}(d)$. Landmark variables are denoted by $\mathbf{L}_n := (L_1, L_2, \ldots, L_n)$ where each element denotes the location or pose of a landmark observed. Let $\mathbf{o}_t := (o_1, o_2, \ldots, o_t)$ be odometry measurements, each modeled by the likelihood function $p(o_i \mid X_{i-1}, X_i)$. Let $\mathbf{z}_t := (z_0, z_1, \ldots, z_t)$ be all landmark measurements, each

Figure 2.4: GAPSLAM image showing relation between the current update of the map being generated to the current video frame processed by front-end. The purple Gaussian appears as an ellipsoid on the map and sphere in the video, while the orange particle filter has a cone shape on the map while having a more rounded appearance in the video. The difference comes from the map being presented from a bird's-eye view downwards perspective while the video is recorded from a human-eye level forward facing perspective.

modeled by the likelihood function $p(z_i \mid X_i, Ld_i)$, where $d_i \in 1, 2, \ldots, n$ is the landmark index associated with measurement z_i. Let $\mathbf{d}_t := (d_0, d_1, \ldots, d_t)$ be data associations for all landmark measurements. Thus, the posterior of robot and landmark variables at time step t given in equation (2.4) can be further developed by applying the Bayes Theorem [88] and assuming conditional independence of measurements given the robot and landmark states, which allows the factorization of the likelihood into the product of individual measurement likelihoods depicted in equation (2.5), where $p(\mathbf{X}, \mathbf{L})$ denotes the prior.

$$p(\mathbf{X}_t, \mathbf{L}_n \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) \tag{2.4}$$

$$\overset{\text{Bayes}}{\propto} \prod_{i=0}^{t} p(z_i \mid X_i, L_{d_i}) \prod_{i=1}^{t} p(o_i \mid X_{i-1}, X_i) p(\mathbf{X}, \mathbf{L}), \tag{2.5}$$

GAPSLAM denotes the GA of the posterior at time step t as $g(\mathbf{X}_t, \mathbf{L}_n \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t)$. We categorize all landmarks into a set of NG landmarks $L_{\mathcal{N}_t}$ and a set of Gaussian landmarks $L_{\mathcal{G}_t} = \mathbf{L}_n \backslash L_{\mathcal{N}_t}$, where $\mathcal{N}_t \cup \mathcal{G}_t = \{1, 2, \ldots, n\}$ and $\mathcal{N}_t \cap \mathcal{G}_t = \emptyset$. We use the GA to represent the posterior of robot poses and the Gaussian landmarks, and the Conditional Independence Relation (CIR) resulting in:

$$p(\mathbf{X}_t, \mathbf{L}_n \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) \tag{2.6}$$

$$\overset{\text{GA}}{\approx} p(L_{\mathcal{N}_t} \mid \mathbf{X}_t, L_{\mathcal{G}_t}, \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) g(\mathbf{X}_t, L_{\mathcal{G}_t} \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t), \tag{2.7}$$

$$\overset{\text{CIR}}{=} p(L_{\mathcal{N}_t} \mid \mathbf{X}_t, \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) g(\mathbf{X}_t, L_{\mathcal{G}_t} \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t), \tag{2.8}$$

$$\overset{\text{CIR}}{=} \prod_{i \in \mathcal{N}_t} p(L_i \mid X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}) g(\mathbf{X}_t, L_{\mathcal{G}_t} \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t). \tag{2.9}$$

The GA can be computed via the Laplace approximation [89]; finding the MAP estimate as the mean and estimating the inverse of the Hessian of the negative logarithm of the posterior at the MAP estimate as the covariance. In practice, many non-linear local optimization solvers, in this case GTSAM, will be used as an effective out-of-the-box solution for the GA.

2.3.2 Marginal Posterior of Non-Gaussian Landmarks

GAPSLAM uses particles to represent the marginal posterior of any NG landmark $L_i \in L_{\mathcal{N}_t}$. Integrating out all variables except L_i in (2.9), we can derive the NG marginal posterior given in equation (2.11). Equation (2.12) is significant because it reveals how the GA side of GAPSLAM contributes to the NG side of GAPSLAM.

$$p\left(L_i \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t\right) \tag{2.10}$$

$$\overset{\text{GA}}{\approx} \int_{X_{\mathcal{V}_i}} p\left(L_i \mid X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}\right) g\left(X_{\mathcal{V}_i} \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t\right) \tag{2.11}$$

$$\overset{\text{MC}}{\approx} \frac{1}{K} \sum_{k=1}^{K} p\left(L_i \mid X_{\mathcal{V}_i} = x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}\right), \tag{2.12}$$

Monte Carlo (MC) integration [90] is applied in (2.12), where $\left\{x_{\mathcal{V}_i}^{(k)}\right\}_{k=1}^{K}$ are K independent and identically distributed samples of part of the robot path drawn from the Gaussian marginal $g\left(X_{\mathcal{V}_i} \mid \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t\right)$. The conditional probability of L_i in equation (2.12) is a result of eliminating L_i from factors adjacent to L_i. Equation (2.12) is further developed by equations (2.13) and (2.14), which describe the key components of the probabilistic model used to estimate landmark positions in a SLAM context. Equation (2.13) defines the conditional probability of a landmark L_i given the relevant robot poses, measurements, and data associations. This probability is expressed as a normalized version of a function ψ, which encapsulates all the information we have about the landmark. Equation (2.14) then defines this function ψ in detail. It is constructed as the product of two main components: first, the likelihood of all measurements related to the landmark, which is represented as a product of individual measurement probabilities for each observation of the landmark; and second, the prior probability of the landmark's position. This formulation allows the integration of both the information gained from sensor measurements and any prior knowledge about landmark positions into a single, coherent probabilistic model. By using this model, the algorithm can estimate landmark positions while accounting for uncertainties in both the robot's trajectory and the sensor measurements.

$$p\left(L_i \mid X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}\right) = \frac{\psi\left(L_i, X_{\mathcal{V}_i}\right)}{\int_{L_i} \psi\left(L_i, X_{\mathcal{V}_i}\right)} \tag{2.13}$$

$$\psi\left(L_i, X_{\mathcal{V}_i}\right) = \prod_{j \in \mathcal{V}_i} p\left(z_j \mid X_j, L_i\right) p\left(L_i\right) \tag{2.14}$$

GAPSLAM uses importance sampling to draw samples representing $p\left(L_i \mid X_{\mathcal{V}_i} = x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}\right)$. We design the proposal distribution of landmark L_i as the sum-mixture of binary factors, as

seen in equation (2.15). with proposal samples $\left\{l_i^{(m,k)}\right\}_{m=1}^{M}$ in hand, the normalized weight of each sample can be computed by equation (2.16), where $\sum_{m=1}^{M} w^{(m,k)} = 1$. Finally, we apply the resampling and regularization steps to generate equally weighted samples of L_i. Thus, through equations (2.17) and (2.18) the marginal posterior in (2.12) can be approximated.

$$q\left(L_i; x_{\mathcal{V}_i}^{(k)}\right) = \frac{1}{|\mathcal{V}_i|} \sum_{j \in \mathcal{V}_i} p\left(L_i \mid X_j = x_j^{(k)}, z_j\right) \tag{2.15}$$

$$w^{(m,k)} \propto \frac{p(l_i^{(m,k)} \mid x_{V_i}^{(k)}, z_{V_i}, d_{V_i})}{q(l_i^{(m,k)}; x_{V_i}^{(k)})} \propto \frac{\psi(l_i^{(m,k)}, x_{V_i}^{(k)})}{q(l_i^{(m,k)}; x_{V_i}^{(k)})}, \tag{2.16}$$

$$p(L_i \mid z_t, o_t, d_t) \approx \tilde{p}(L_i \mid z_t, o_t, d_t) \tag{2.17}$$

$$\tilde{p}(L_i \mid z_t, o_t, d_t) = \frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \delta(L_i - l_i^{(m,k)}) \tag{2.18}$$

To summarize, for each of the K robot path samples, we draw M landmark samples. The delta functions (or other kernels) of all KM landmark samples can be used to approximate the marginal posterior of the landmark.

2.3.3 Reinitialization

From time step t to time step $t+1$, the robot receives a new odometry reading o_{t+1} and a new landmark measurement z_{t+1}. It is easy to create a new robot pose X_{t+1} and absorb the odometry o_{t+1} to form the intermediate Gaussian $g\left(\mathbf{X}_{t+1}, \mathbf{L}_n \mid \mathbf{z}_t, \mathbf{o}_{t+1}, \mathbf{d}_t\right)$. However, fusing the measurement z_{t+1} is a more involved process. Assuming that the measurement z_{t+1} comes from a landmark with index d_{t+1}, several cases are considered:

1. If the observed landmark $L_{d_{t+1}}$ is not new or a NG landmark, we simply add the likelihood model of z_{t+1} the factor to the Gaussian solver to update the Gaussian approximation without making any further changes.

2. If the observed landmark $L_{d_{t+1}}$ is new, i.e., $d_{t+1} = n+1$, we update the landmark set to $\mathbf{L}_{n+1} = \mathbf{L}_n \cup \{L_{n+1}\}$ and indices of NG landmarks to $\mathcal{N}_{t+1} = \mathcal{N}_t \cup \{n+1\}$. We then add the likelihood model of measurement z_{t+1} to the Gaussian solver. Finally, we can simulate equally weighted samples of the landmark using this single measurement and randomly select a point from the samples as the initial value of the landmark.

3. If the observed landmark $L_{d_{t+1}}$ is not new and is in NG landmarks $L_{\mathcal{N}_t}$, reinitialization must be considered.

In the third case, our goal is to use the new measurement z_{t+1} to determine whether we should explicitly reinitialize the NG landmark in the Gaussian solver. Furthermore, if our belief of the landmark converges to a less uncertain situation after fusing measurement

z_{t+1}, we remove the landmark from the NG set $L_{\mathcal{N}_t}$, recording the landmark as being purely Gaussian.

The union of the samples and the current estimate of the landmark forms a set of candidate points for reinitialization. The point in the set that maximizes the posterior will be used to reinitialize the landmark. During the evaluation of the posterior, all variables other than the landmark are fixed by the current mean of the GA. This reduces the evaluation to only the product of factors that are adjacent to the landmark, simplifying the computation. The reinitialization formula is given in equation (2.19).

$$l_i^* = \underset{l_i \in \{\mathcal{S}, \hat{l}_i\}}{\operatorname{argmax}} \psi \left(L_i = l_i, X_{\mathcal{V}_i} = \hat{x}_{\mathcal{V}_i} \right) \tag{2.19}$$

When the largest eigenvalue of the NG landmark's covariance matrix falls below a threshold, it indicates that the belief of landmark L_i has become sufficiently certain, and L_i is removed from the NG landmark set $L_{\mathcal{N}_{t+1}}$. As a result, L_i will no longer be estimated by particle filters and will only be involved in updates in the Gaussian solver, giving GAPSLAM enough confidence in the landmark that it draws it as a GA ellipsoid on the map.

2.3.4 GAPSLAM Data Association

GAPSLAM leverages the following five heuristics with respect to its data association scheme, which assist in determining candidates for reinitialization:

1. Intersection is not allowed. The system removes all but one of multiple detections with the same object classes intersecting each other on the camera frame.

2. Data association for detections of the same classes within proximity of each other is determined using Mahalanobis distance [91]. The threshold is the critical value of the chi-square distribution [92] with a 99 percent confidence level with two degrees of freedom for simplicity in testing for goodness-of-fit, which has approximately a value of 9.21.

3. Rejection of detections of different classes within proximity of each other is determined using Mahalanobis distance. The threshold is the same value as above.

4. Sample-based data association for NG landmarks of the same class is accomplished by projecting landmark samples to the image and checking the percentage of samples within the bounding box. The default threshold is 10 percent of the samples.

5. New landmark factor is generated if there are no associations to prior landmarks.

2.4 Object Detectors

Objector detectors are an essential aspect of GAPSLAM as they are required for the creation of observation factors in the front-end of GAPSLAM. The semantic distinction of their object classes can be leveraged to filter out transient moving objects such as marine fauna, by simply discarding detections of known moving objects as observation factors. Additionally,

semantic information carried by object detectors is essential for marine autonomy as it imbues meaning onto maps created by SLAM. With a semantic map, the AUV is able to distinguish which parts of the map are useful for achieving its objectives. For example, in the pipeline inspection problem, object detectors and SLAM can be leveraged to detect and map where corrosion has appeared relative to various points of reference on the pipeline.

2.4.1 Detic

Detic (Detection Transformer for Instance-level Recognition) [93] is a SOTA object detection framework that builds on the principles of the Transformer [94] architecture, which has revolutionized natural language processing tasks. The core of the Transformer architecture is the self-attention mechanism, which allows the model to focus on different parts of the input sequence when making predictions. In the context of Detic, this helps in focusing on different regions of an image to identify objects. Detic utilizes the encoder-decoder structure where the encoder processes the input image, and the decoder generates predictions about the objects present in the image. Unlike traditional methods like Faster Region-Based Convolutional Neural Network (R-CNN) [95] that generate a large number of proposals and refine them, Detic performs object detection as a direct set prediction problem, predicting a fixed number of object instances directly from the image. Additionally, Detic provides instance-level segmentation masks, distinguishing between different instances of the same object class, such as two different dogs in the same image. To ensure that each predicted object corresponds to a ground truth object, Detic employs a bipartite matching loss that considers both the class of the object and its spatial location. The input image in Detic is divided into a grid of patches, each flattened and linearly embedded into a feature vector, with positional encodings added to retain spatial information. The Transformer encoder processes these embedded patches using multi-head self-attention layers and feed-forward neural networks, capturing global context and relationships between different parts of the image. A fixed set of learnable object queries is used in the decoder, representing potential objects in the image and interacting with the encoded image features through cross-attention mechanisms. Finally, the decoder outputs are processed by prediction heads to generate final detections, including the class of the object, bounding box coordinates, and segmentation masks.

2.4.2 YOLO

YOLO (You Only Look Once) [96] is a family of Convolutional Neural Network (CNN) models designed for real-time object detection. Unlike traditional methods that use a region proposal-based approach, YOLO reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. YOLO employs a single neural network to predict bounding boxes and class probabilities directly from full images in one evaluation, a single-stage approach that contrasts with the two-stage methods used by R-CNN, which first generate region proposals and then classify them. The input image is divided into an $S \times S$ grid, with each grid cell responsible for predicting a fixed number of bounding boxes and their associated confidence scores, as well as the probability distribution over classes. Each bounding box is defined by four coordinates (x, y, width,

height), a confidence score indicating the likelihood that the box contains an object, and the class probabilities. The confidence score reflects the accuracy of the bounding box and whether it contains an object, calculated as the intersection-over-union between the predicted box and the ground truth box. Each grid cell also predicts the conditional class probabilities for the object if the object center falls into that cell. YOLO's faster models are particularly suitable for real-time SLAM problems.

2.5 Summary

This chapter has reviewed the formulation of the visual SLAM problem and discussed the critical components of visual semantic SLAM systems, spanning the front-end and back-end. We have reviewed past work in both underwater visual SLAM and semantic SLAM and posed our goal of underwater semantic SLAM using the GAPSLAM algorithm developed by Huang and Leonard [87]. We also reviewed the two primary object detectors that we will use in our work: Detic and YOLO. In the next chapter, we will evaluate GAPSLAM in a terrestrial environment before discussing our new real-time underwater implementation and experiments in the following chapters.

Chapter 3

Preliminary Study

Before embarking on the adaptation and evaluation of GAPSLAM in an underwater environment, we first present the results from a terrestrial implementation in a small lab that uses a motion capture system for ground truth.

3.1 Evaluation of SLAM

3.1.1 Ground Truth

In order to benchmark and measure how effective GAPSLAM is in mapping objects and the robot's current state with reference to the map, an absolute reference frame, referred to as GT, is required. The most common way of comparing different SLAM systems is to compare the final trajectories to the ground truth trajectory for a given dataset. This is because a trajectory is the collection of the localized robot state through time, which is accomplished simultaneously with the mapping of objects from observations made from the current robot pose. Therefore, the two cannot be separated and a more accurate trajectory equates to a more accurate map as well [97].

Trajectory comparisons for GAPSLAM were presented by Huang [87] for a large lab/office environment. However, there is room to improve regarding this experiment as it uses pseudo-GT from a single loop of ORB-SLAM3, which is not ideal. While it is known how accurate ORB-SLAM3 is from previous experiments with GT, using ORB-SLAM3 as pseudo-GT for GAPSLAM is ultimately validating an estimate with another estimate that was run under different conditions and comes with risks. It can be useful to get a rough estimate for comparable feature-based SLAM methods when running experiments similar to the one that validated ORB-SLAM3 with GT. However, when comparing GAPSLAM to ORB-SLAM3 it is expected that there is a possibility of non-negligible deviation for certain experimental setups, because while ORB-SLAM3 will always have sufficient features to work with in an office environment, that is not the case for GAPSLAM, where admitted object classes, quantity and their specific layout in the space can greatly affect performance.

Therefore, simultaneous GT testing for GAPSLAM and ORB-SLAM3 was planned. The twelve low-latency camera Vicon motion capture system [98] at MIT building 32-339 was used to compare GAPSLAM to ORB-SLAM3 and the ground truth. A Vicon motion capture

system is considered to be high-performance and tracks object based on markers. In a typical lab setup, three or more markers are attached in a unique geometric configuration to each rigid-body object for identification. Multiple cameras that are sensitive to the reflective rays from the markers are mounted to monitor a capture volume. A host computer is connected to all cameras processing reflective ray detection. The camera poses and absolute scale of the real world can be determined by calibration algorithms. With calibrated cameras, the three-dimensional position of a marked object can be determined if the object is in view of multiple cameras at the same time. A typical Vicon system setup is capable of determining position with millimeter accuracy and with a measurement update frequency greater than 100 Hz.

3.1.2 EVO

Ground truth must not only be the most accurate method but its data must also have accurate timestamps. While compared trajectories do not need to have identical timestamps or the same number of datapoints, timestamps between different compared trajectories must share the same reference point with regards to time. This is because different SLAM systems and methods for determining trajectory often have different spatial reference points and different scales or multipliers, and the transformation matrices required to unify the trajectory reference points are typically unknown. Therefore, time is often the only common reference point that allow legitimate comparison.

Evaluation of Odometry and SLAM (EVO) [99], is a framework that uses timestamps to compare various trajectories from the same dataset. It supports TUM [100], KITTI [101], EuRoC [82] and Robot Operating System (ROS) bag files [102]. For the GAPSLAM evaluations, TUM format was used, which contains eight columns of data: timestamp, position (x, y, z), and quaternion rotation $(qx, qy, qz, andqw)$. EVO is useful because it can visualize the plotting of multiple trajectories, as well as quantifying the APE between trajectories. EVO has options to align trajectories and correct scales using Sim(3) Umeyama [74] [75] alignment of rotation, translation and scale. One of the most important parameters in EVO for APE and trajectory comparison is the *t_ max_ diff*. An optimal value is required for this parameter because if it is set too low, then only a small fraction of the compared trajectory's data points will be used in the comparison, yielding drastically different results. Too high of a value increases the chances of irrelevant outliers being included due to excessive lack of constraints allowed in comparison. The *t_ max_ diff* parameter was tune so that at least 95 percent of the data points were used in the comparison.

3.2 Vicon Room Experiment

ORB-SLAM3 was compared to both Detic GAPSLAM and YOLOv8 GAPSLAM across three runs, where each run consisted of three Vicon reflectors rigidly attached to an Intel D435i camera, which did a full loop around the Vicon room using a manual push cart as a platform. Essentially, the camera was mounted to a flexible hinge that allowed pitch angle changes and z-axis motion because it was fitted on the backside of the laptop's screen. The laptop was on the pushcart, acting as the server that was timestamping and storing the Vicon

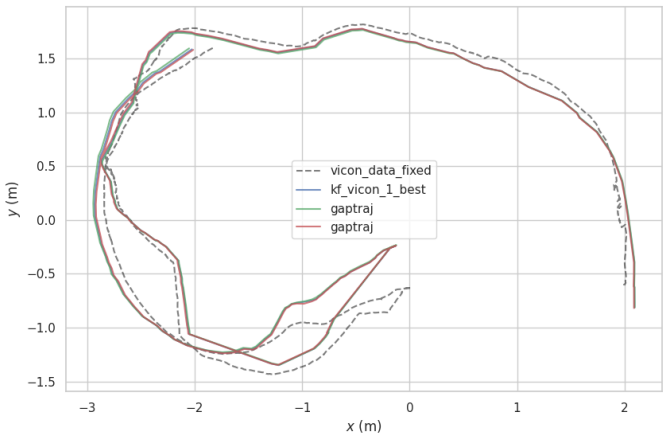


Figure 3.1: Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 1 dataset.

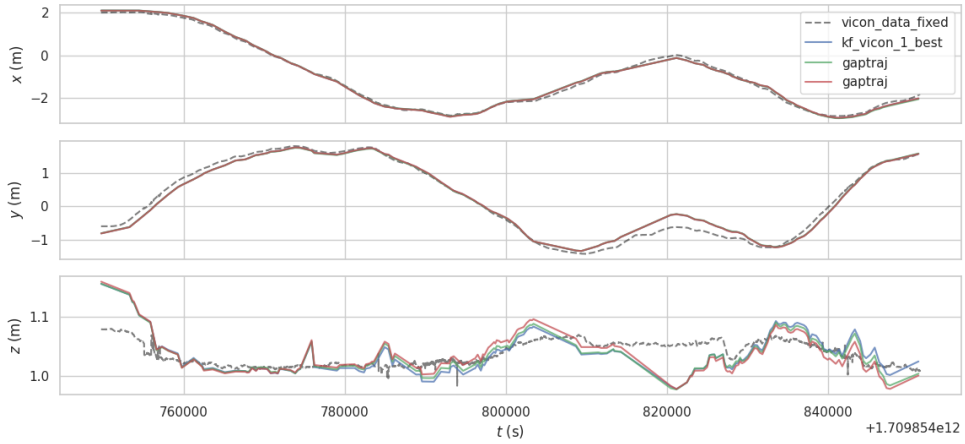


Figure 3.2: $x - y - z$ Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 1 dataset.

Metric	Detic GAPSLAM	YOLOv8 GAPSLAM	ORB-SLAM3
max	0.417	0.417	0.420
mean	0.143	0.144	0.144
median	0.129	0.131	0.132
min	0.016	0.023	0.023
rmse	0.166	0.167	0.168
sse	4.339	4.385	4.409
std	0.085	0.085	0.086

Table 3.1: Best APE results w.r.t. translation part (m) with Sim(3) Umeyama alignment for Detic GAPSLAM, YOLOv8 GAPSLAM and ORB-SLAM3 on Vicon 1 dataset.

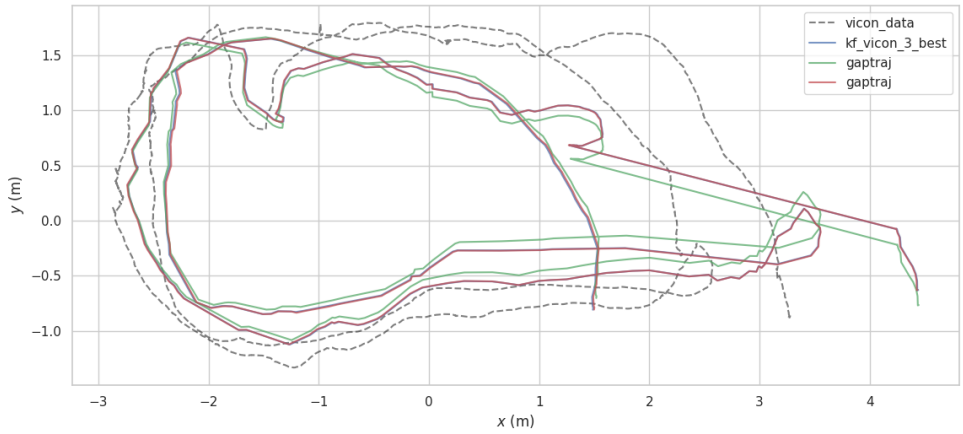


Figure 3.3: Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 2 dataset.

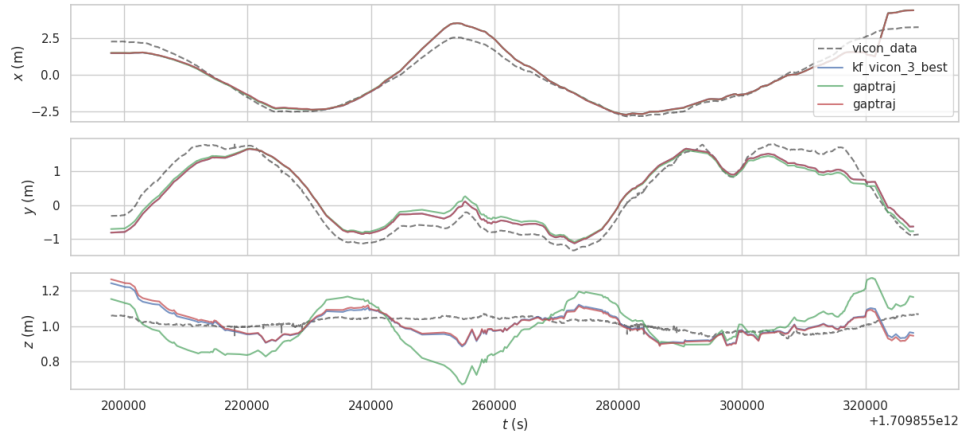


Figure 3.4: $x - y - z$ Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 2 dataset.

Metric	**Detic GAPSLAM**	**YOLOv8 GAPSLAM**	**ORB-SLAM3**
max	1.664	1.673	1.680
mean	0.577	0.574	0.577
median	0.415	0.409	0.416
min	0.074	0.071	0.078
rmse	0.686	0.683	0.686
sse	123.165	122.060	123.165
std	0.369	0.369	0.370

Table 3.2: Best APE results w.r.t. translation part (m) with Sim(3) Umeyama alignment for Detic GAPSLAM, YOLOv8 GAPSLAM and ORB-SLAM3 on Vicon 2 dataset.

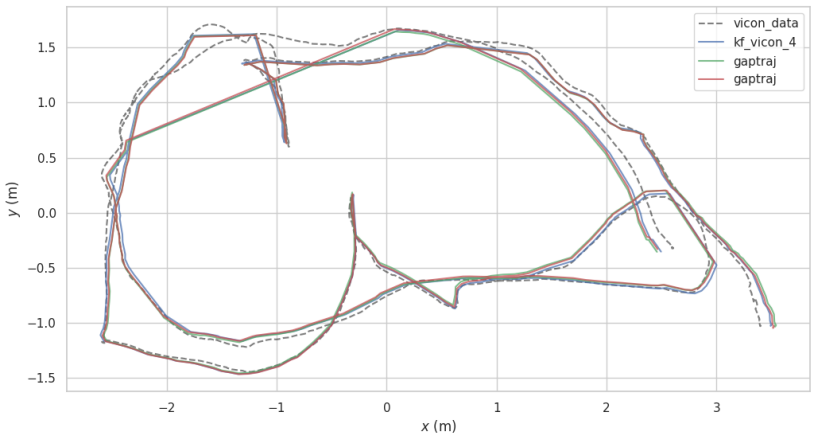


Figure 3.5: Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 3 dataset.

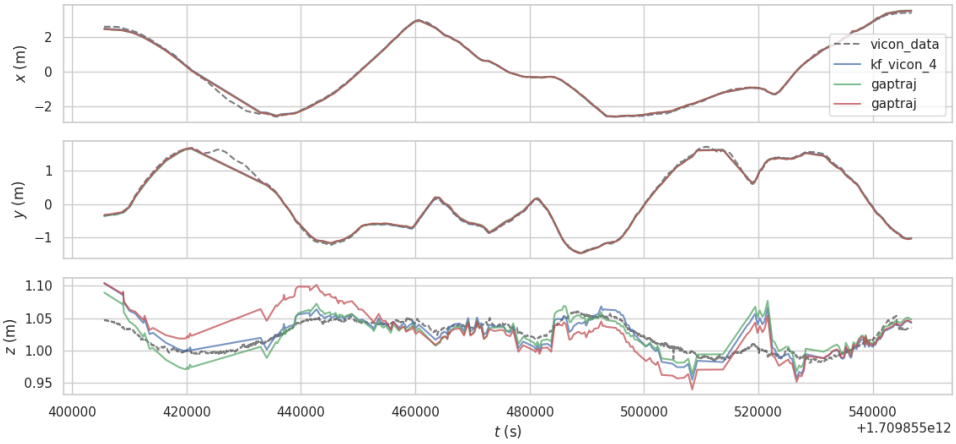


Figure 3.6: $x - y - z$ Comparison of Detic (red) and YOLO (green) GAPSLAM and ORB-SLAM3 (blue) against the Vicon motion capture system ground truth (black slotted) trajectory on Vicon 3 dataset.

Metric	Detic GAPSLAM	YOLOv8 GAPSLAM	ORB-SLAM3
max	0.158	0.146	0.125
mean	0.048	0.055	0.044
median	0.037	0.045	0.041
min	0.002	0.003	0.004
rmse	0.057	0.062	0.051
sse	0.820	0.979	0.648
std	0.029	0.027	0.024

Table 3.3: Best APE results w.r.t. translation part (m) with Sim(3) Umeyama alignment for Detic GAPSLAM, YOLOv8 GAPSLAM and ORB-SLAM3 on Vicon 3 dataset.

pose data via Ethernet as well as the D435i video frames via Universal Serial Bus (USB). The same keyframes were used for both GAPSLAM and ORB-SLAM3 to ensure a direct comparison, and each run was relatively accurate to GT with only a minor APE difference between the different methods. On the Vicon 2 dataset 3.2 however, the loop closure was not detected, resulting in more significant APEs for ORB-SLAM3, and consequently both Detic and YOLO GAPSLAM as well. One unexpected observation was that a significant portion of the APE originated from the z-axis as shown in figures 3.2, 3.4, and 3.6. A small portion of this APE is likely due to the flexible hinge that allowed vibration from the small bumps on the floor to produce a number of blurry images in the relatively low 30 FPS camera. The most significant source of z-axis APE is likely due to insufficient constraints due to less opportunities for loop closures to impact the z-axis as the motion predominantly occurred in the x-y plane.

Metric	**Detic 0.60**	**Detic 0.65**	**Detic 0.70**	**Detic 0.75**	**Detic 0.80**	**Detic 0.85**	**Detic 0.90**
max	-	0.158	0.174	0.153	0.173	0.163	0.160
mean	-	0.052	0.054	0.056	0.062	0.049	0.048
median	-	0.045	0.043	0.051	0.058	0.037	0.040
min	-	0.006	0.003	0.005	0.005	0.006	0.002
rmse	-	0.060	0.064	0.063	0.070	0.059	0.057
sse	-	0.920	1.055	1.019	1.227	0.884	0.820
std	-	0.030	0.035	0.030	0.031	0.033	0.031
Total Run Time (s)	-	12.760	10.160	9.350	7.610	6.470	5.910
Total Objects	-	139	105	74	46	21	10
Gaussian-Only Objects	-	31	14	14	8	3	1
Non-Gaussian Objects	-	108	91	60	38	18	9
Gaussian Ratio	-	0.223	0.133	0.189	0.174	0.143	0.100

Table 3.4: APE w.r.t. translation part (m) with Sim(3) Umeyama alignment for Detic GAPSLAM on Vicon 3 dataset across different detection score filters. Object detection score filter of 0.60 failed for default Detic GAPSLAM settings, so it was omitted. It does successfully run however, if running parameters are slightly tuned.

Metric	**YOLO 0.60**	**YOLO 0.65**	**YOLO 0.70**	**YOLO 0.75**	**YOLO 0.80**	**YOLO 0.85**	**YOLO 0.90**
max	0.174	0.162	0.159	0.150	0.149	0.146	0.151
mean	0.055	0.056	0.058	0.056	0.057	0.056	0.055
median	0.045	0.048	0.049	0.049	0.050	0.050	0.048
min	0.005	0.008	0.007	0.007	0.005	0.003	0.007
rmse	0.064	0.063	0.065	0.062	0.063	0.063	0.062
sse	1.056	1.018	1.078	0.987	1.011	0.993	0.979
std	0.033	0.030	0.030	0.028	0.027	0.027	0.028
Processing Time (s)	2.10	3.57	2.50	2.86	2.55	2.37	1.99
Total Objects	32	29	24	22	15	10	7
Gaussian-Only Objects	4	3	2	2	4	3	0
Non-Gaussian Objects	28	26	22	20	11	7	7
Gaussian Ratio	0.125	0.103	0.083	0.091	0.267	0.300	0.000

Table 3.5: APE w.r.t. translation part (m) with Sim(3) Umeyama alignment for YOLOv8 GAPSLAM on Vicon 3 dataset across different detection score filters.

Unexpectedly, there was no significant APE difference between YOLO and Detic GAPSLAM as shown in the Vicon 1 results table 3.1 and the Vicon 2 results table 3.2. The dataset that resulted in the biggest difference between Detic and YOLO was Vicon 3 3.3, which was also

the dataset that has the lowest mean APE with ORB-SLAM3 having even lower mean APE. Comparison between YOLO and Detic across various object detection score filters is shown in tables 3.4 and 3.5. While Detic used the default unfiltered Large Vocabulary Instance Segmentation (LVIS) [103] list of 1024 object classes the default YOLOv8 model was trained on 80 classes, which meant that Detic was guaranteed to detect a much larger number of objects given the same minimally acceptable object detection score filter.

Because the VO in this run was highly accurate as shown in the ORB-SLAM3 results for the Vicon 3 results table 3.3, having a large number of total objects did not directly correlate to overall accuracy, as there were other performance metrics that were impaired. Aside from taking a longer duration to process frames, Detic having a large list of object classes disadvantages precision and recall, where there is an increased chance for false positives and false negatives during detection. Certain objects fall within multiple detection classes due to similarity in appearance or function. For example, Detic in the Vicon room often mistakes the trash can as the bucket class, and the Lego class gets mistaken for the toy class. This multiple interpretation is shown in figure 3.9, where the only consistent detections across all three frames is the purple particle filter inside the box on the left, although there is only a small amount of rotation and translation between frames, especiall from the left frame to middle frame.

Although object detections may be semantically similar or equivalent, this alternate interpretation of the objects in the frames could result in a transient new landmark of a different class that is only observed once. When this occurs, it adversely affects optimization by denying a potential loop closure, delaying GAPSLAM reinitialization as well as maintaining an erroneous observation in the factor graph by being detected as a different class from its previous frame, potentially alternating classes from one frame to another. Objects being comprised of component objects such as the chair class being further divided into cushion class could also potentially contribute to this problem.

Although not a direct correlation, a metric of importance in the Vicon 3 dataset was the Gaussian Ratio, which was the ratio of reinitialized Gaussian-only objects relative to the total number of detected objects. This was expected because Gaussian-only landmarks indicate great confidence in the observation. Generally, a higher Gaussian Ratio resulted in better max and mean APE. Vicon 3 runs that performed well that did not have a good Gaussian Ratio had a small number of total detections instead, which was also an expected result as a small number of total objects meant there was a smaller chance of transient erroneous observations as described above.

3.2.1 Differences from Original GAPSLAM Experiment

ORB-SLAM3 outperformed GAPSLAM in the Vicon 3 dataset, which was not the case for the results of the original GAPSLAM experiment (shown in [87]). The main cause of GAPSLAM outperforming ORB-SLAM3 in the original experiment was that ORB-SLAM3 was unable to detect the loop closure after three loops of the experiment environment. In the case of our experiment where the loop closure is detected for both GAPSLAM and ORB-SLAM3 in the Vicon 3 dataset, ORB-SLAM3 has a slight advantage in accuracy as it uses over one thousand matched ORB features while GAPSLAM only relies on a handful of landmarks as shown in table 3.3. Additionally, whereas [87] used only nine classes of objects

Figure 3.7: Semantic map of objects and trajectory of YOLO and Detic GAPSLAM produced in the Vicon 3 dataset for an object detection score of 0.65. Object classes are not labeled in the figure, but is recorded and can be retrieved.

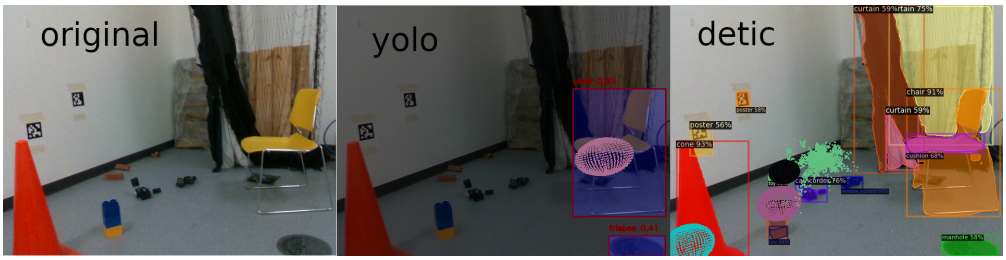


Figure 3.8: Frame 136 of the Vicon 3 dataset. with original on the left, YOLO without segmentation in the middle, and Detic with segmentation on the right.

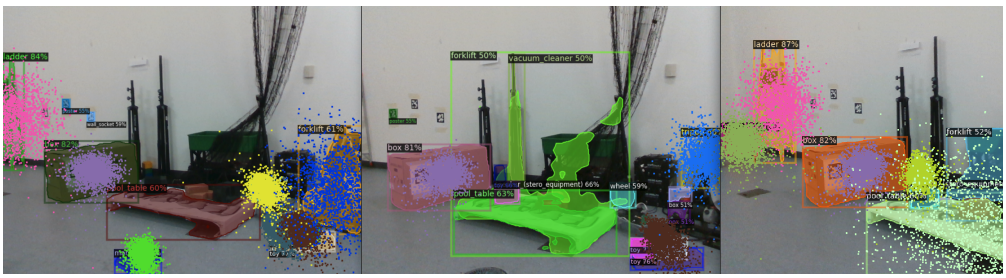


Figure 3.9: Comparison of adjacent frames processed by Detic in the Vicon 1 dataset.

(cup, cereal box, trash can, skateboard, office chair, football, bottle, traffic cone, and toy car) that had strong recall rates resulting in a reduced number of false positive object detections, the Vicon room experiments did not use any filters for any unfavorable classes. This further increased refinement for [87] compared to the map and trajectory generated in the Vicon room.

3.3 Summary

Having evaluated our GAPSLAM system in a lab environment with motion capture, in the next chapter, we proceed to discuss the integration of GAPSLAM with the software system of the BlueROV2 underwater vehicle.

Chapter 4

Methodology

This chapter describes our online underwater object-based SLAM system for the BlueROV2. The overall system achitecture is depicted in Figure 4.1. The system uses ROS to communicate with the vehicle sensors and Redis to communicate with the graphical processor as well as to make the real-time visualization available.

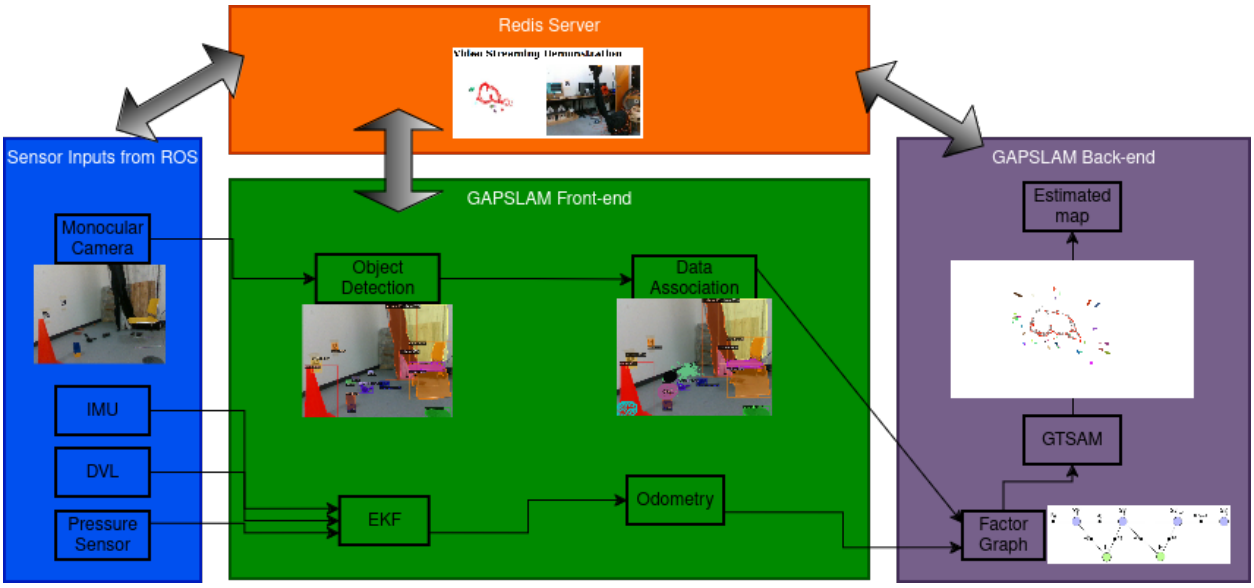
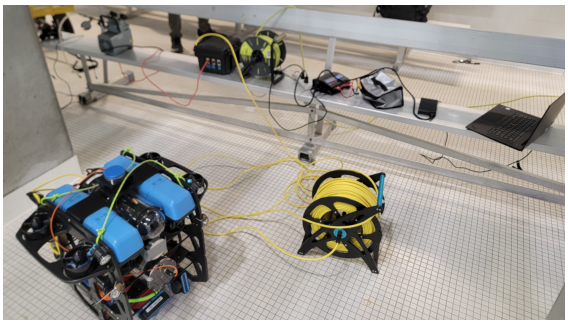


Figure 4.1: System architecture of online GAPSLAM for the pool experiment. The thin arrows show the procedural flow of data of GAPSLAM, while the thick arrows show the dependency of GAPSLAM on Redis for each transfer of data from one module to another.

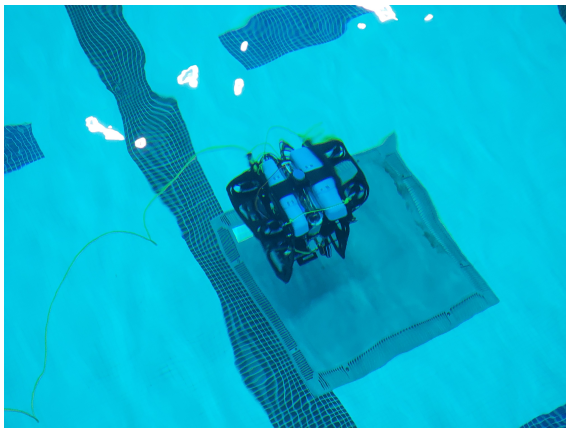
4.1 Hardware

The platform chosen to test underwater GAPSLAM was BlueROV2, which is one of the ROV assets of MIT's Marine Robotics Group. Although the ideal vehicle to implement underwater GAPSLAM should be a tether-free AUV in order to allow the vehicle more potential to be autonomous, commercially available AUVs [104] are generally equipped with a processor

with frequency of 0.1 to 1.0 GHz that cannot handle the significant graphical requirements of real-time object detection for images that have hundreds of pixels for width and height in file size. This is because graphics-intensive tasks has traditionally not been required on commercial AUVs, and demand has only appeared in cutting-edge research projects in very recent years at the time of writing. Therefore, for this experiment the BlueROV2 was equipped with a tether that connects to a laptop with a dedicated graphical processor. Essentially, the BlueROV2's Raspberry Pi 4 [105] is only used to network the vehicle's sensors to the laptop's processors via ROS. The laptop used for front-end graphics processing and back-end factor graph optimization is the Maingear Vector Pro 2, which is equipped with an Intel i9-12900H Central Processing Unit and an NVIDIA RTX 3080Ti Mobile Graphics Processing Unit.



(a) The MIT Marine Robotics Group's BlueROV2 setup, including laptop and extension cable.



(b) The MIT Marine Robotics Group's BlueROV2 present at the Z-Center pool.

Figure 4.2: MIT Marine Robotics Group's BlueROV2 and setup.

4.2 Robot Operating System

The BlueROV2 [106] has on board the BlueOS Linux operating system, which makes ROS the interface of choice when communicating with the laptop. ROS is ideal for running experiments with GAPSLAM on real robots because on board sensor data can be linked to ROS topics, which can be configured to automatically transmit a timestamped message containing sensor data when data becomes available. The timestamps are reliably measured in microseconds, and is a common point of reference for all ROS topics, which is essential in sensor fusion and setups where there are multiple machines working together across the same network.

4.3 Redis Server

Redis [107] is an in-memory data structure store that serves as a database, cache, and message broker. It supports a wide array of data structures such as strings, hashes, lists, sets, sorted

sets with range queries, bitmaps, hyperlogs, geospatial indexes, and streams. The primary advantage of Redis lies in its in-memory storage capability, which enables extremely fast read and write operations, essential for applications where low latency and high throughput are critical. This attribute makes Redis particularly suitable for real-time applications like GAPSLAM, where it can efficiently handle the rapid transfer of video frames.

Moreover, Redis offers a diverse set of data structures. These include binary-safe strings that can store any data type, hashes for object storage, lists which are linked lists of strings, sets which are unordered collections of unique strings, and sorted sets that allow sorting of elements based on scores. Redis also provides specialized data structures such as bitmaps, HyperLogs, geospatial indexes, and streams for specific use cases. Redis also incorporates a publish/subscribe messaging paradigm, where messages are published to channels, and clients subscribe to these channels to receive updates, further enhancing its utility as a message broker.

4.4 EKF Odometry

One of the main problems with adapting GAPSLAM for underwater operations is that VO fails more frequently in underwater environments due to a lack of features. In the indoor pool environment at MIT's Zesiger Center (Z-Center), where the initial underwater evaluation of GAPSLAM was performed, ORB-SLAM3 lost track at 27 percent completion and could not finish the dataset [63]. Similar performance was also observed in Grimaldi's work [106], where most of the tests performed using ORB-SLAM3 without perfect illumination failed. Past experiments where ORB-SLAM3 succeeded were ones where after image enhancement, there was plenty of diverse and robust features throughout the entirety of the dataset. In the case of our indoor pool dataset, there were sections of video between the objects where the camera frames captured only a blurry white pool wall with no discernible features. Even with advanced image enhancement leveraging machine learning or higher resolution cameras, it is not possible to create features that are not there in the first place. Therefore it was determined that ORB-SLAM3 and VO was not suitable for the indoor pool dataset.

In order to avoid failure and have a common reference point for comparing GAPSLAM and OAS-SLAM, we added EKF odometry fusing DVL, IMU and pressure sensor data [63] to GAPSLAM [30], [31]. The IMU provides data on the vehicle's acceleration, orientation, and angular rates, which can be used to estimate its position in x, y and z, and orientation over time. However, due to errors and drift, IMU data alone are usually insufficient for precise navigation.

A DVL measures the velocity of the vehicle relative to the seafloor or water column by using the Doppler effect. The DVL emits acoustic pulses and measures the frequency shift of the echoes reflected from the seabed or suspended particles in the water. The frequency shift is proportional to the velocity of the vehicle, allowing the DVL to calculate the speed in 3D along its four beams oriented at fixed angles. The pressure sensor, often integrated with a temperature sensor, is used to measure the depth of the underwater vehicle by measuring the hydrostatic pressure, which is directly proportional to depth, exerted by the water column above it. The EKF is a state estimation algorithm that combines measurements from its input sensors. The EKF maintains a state vector that includes the vehicle's position, velocity,

orientation, and sensor biases. Using the vehicle's motion model matrix and IMU data, the EKF predicts the current state and its error covariance matrix via differentiation [108]. The EKF updates the state estimate using observations from the DVL and pressure sensor. The measurements are compared to the predicted state, and the discrepancies (residuals) are used to correct the state estimate. This robust sensor fusion reduces trajectory drift from noise accumulation.

While it was possible to pair every EKF output with every image topic message to be processed by GTSAM, doing so overloaded the solver with redundant factors numbering in the thousands, causing the system to become underdetermined and insolvable. There were too many adjacent frames where there was negligible changes going from frame to frame, which caused the system to become less sparse. In ORB-SLAM3, keyframes are selected for efficient map building and optimization, and EKF GAPSLAM required a similar framework. A filter was implemented in order to adapt the number of factors managed by the solver. The ROS message filter was used to match the EKF topic to the camera topic, and a message pair was admitted only when there was both sufficient overlap and change in the next frame. This helped make the solver operate efficiently while keeping the frame rate high enough to be useful for real-time operation. Furthermore, the transformation matrix of equation (4.1) from the EKF odometry's navigation frame to the camera frame was calibrated and were applied to the odometry so that both camera and odometry shared the same frame of reference.

$$T_{nav}^{usb\_cam} = \begin{bmatrix} \cos(1.5708) & -\sin(1.5708) & 0 & -0.01586 \\ \sin(1.5708) & \cos(1.5708) & 0 & 0 \\ 0 & 0 & 1 & 0.1877 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.5 Custom YOLOv8 Model Training

The default model for YOLOv8 only has 80 different object classes, and none of them are objects typically found underwater. Therefore, it was required to train a YOLO model capable of detecting underwater objects. Because underwater GAPSLAM was going to be compared to OAS-SLAM, the YOLO model for underwater GAPSLAM was fine-tuned from the baseline in the same manner as described by Singh et al. [63].

Training a YOLOv8 model on a custom dataset involves several critical steps to ensure precise object detection. Initially, gathering and preparing the dataset is essential, which includes collecting images that represent the target objects. Each image must be annotated with bounding boxes around the objects of interest, specifying the coordinates of the bounding boxes and class labels.

The dataset is then divided into training, validation, and possibly test sets to robustly train the model and evaluate its performance accurately. Setting up the YOLOv8 environment follows, requiring the installation of necessary software and dependencies such as Python, PyTorch, and the YOLOv8 framework.

Configuration of training parameters is carried out in a configuration file. This file includes settings such as paths to the training and validation datasets, the number of classes,

model architecture specifics, learning rates, batch size, and the number of epochs. Fine-tuning these parameters based on the dataset's size and complexity is crucial for optimizing training performance. Monitoring the model's performance through metrics like loss and accuracy during training is vital for making necessary adjustments to the training parameters.

Post-training, evaluating the model's performance on the validation set is essential, and adjusting detection confidence thresholds is necessary to balance precision and recall effectively. The results of this step are shown in figures 4.3 and 4.4. PR is important because it provides a comprehensive view of a model's performance, especially in imbalanced datasets, while F1 is important as it offers a single, balanced metric that considers both precision and recall. Further fine-tuning and retraining may be required based on the evaluation results to achieve desired accuracy and performance, although the results in the figure show that the model is trained to have excellent performance in both precision and recall, with a reasonable trade-off of only two objects out of the thirteen having decent scores instead of excellent scores like the remainder of object classes. The trash-bin and tire classes would be removed from the pool experiment due to their relatively lacking performance.

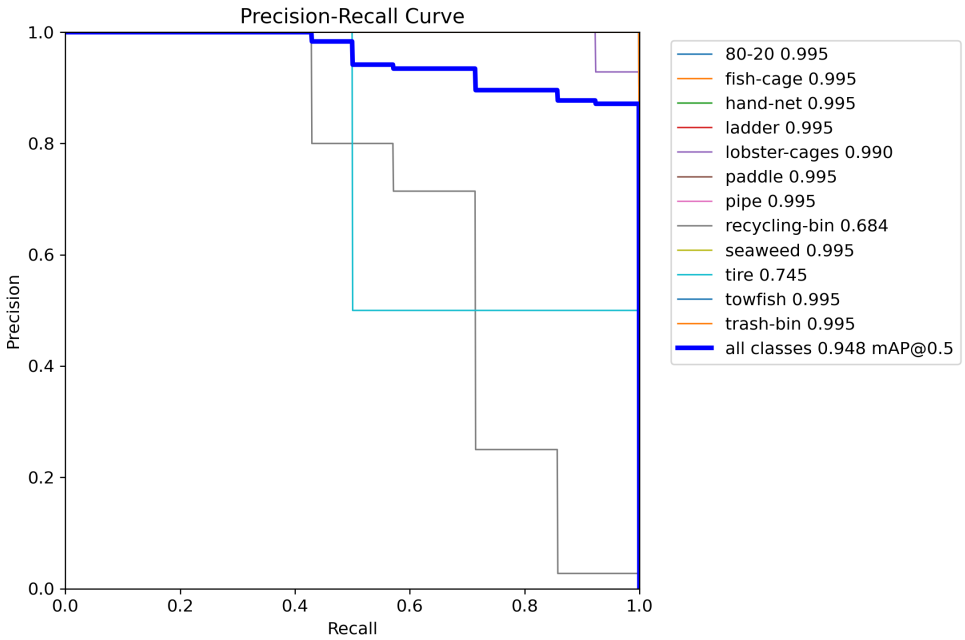


Figure 4.3: PR curve. Other than the trash-bin and tire classes, the object classes have excellent precision and recall.

The above tasks were implemented by using 230 hand-labeled images including every object class in the datasets to fine-tune a YOLOv8 network over 25 epochs. The 230 images were augmented with rotation and blurring to produce a dataset of 691 training images. A sample of test images, that appear highly similar to the target environment, but not identical due to blurring and rotation, is shown in figure 4.5 having detected all of the objects in custom classes it was trained on with high confidence scores. The trained underwater YOLO model also defines the limits for this implementation of underwater GAPSLAM: prior knowledge of the environment and objects of interest must be known in order to operate as expected.

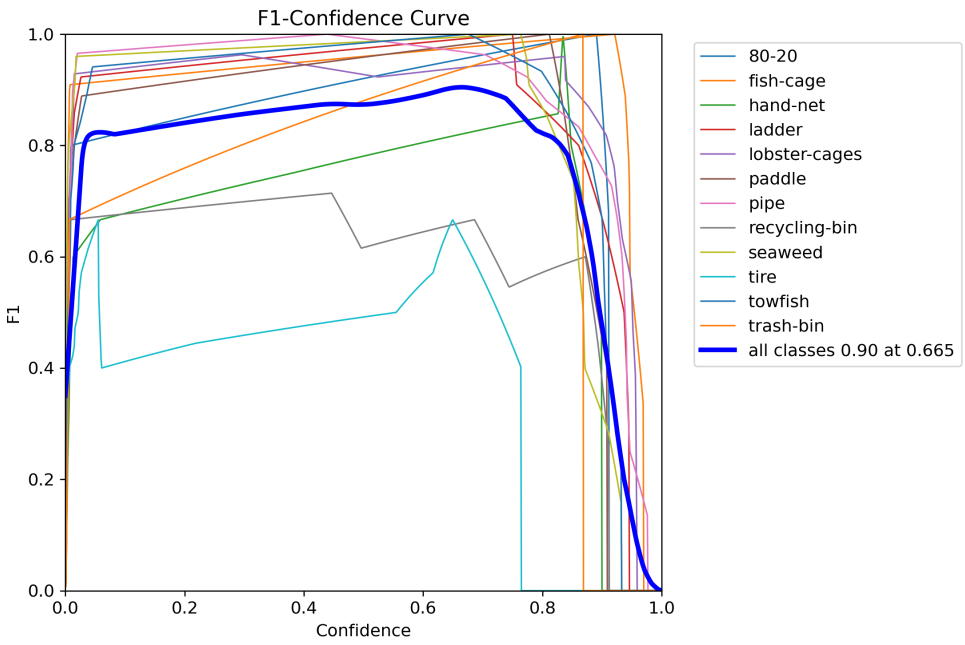


Figure 4.4: F1 Confidence Curve, which shows a balanced metric that considers both precision and recall. It is shown here that the fish-cage and hand-net classes also have below average confidence for specific ranges of F1 scores.

Furthermore, while object detection data for training is easy to label, data for segmentation is more challenging as there is increased annotation complexity. For segmentation tasks, each object in the training images must be annotated with pixel-level precision, requiring manual outlining of the exact shape of each object. This is more time-consuming and intricate compared to bounding box annotations used for detection tasks. Therefore, the underwater YOLO detector was trained only to detect and not segment. This was accepted because it was expected that switching from object segmentation masks to bounding boxes in the front-end would negligibly impact the performance of GAPSLAM. This is because for most objects, the centroid of the object segmentation mask is not significantly different from the centroid of the bounding box.

4.6 Summary

This chapter has described the architecture for our underwater SLAM, discussed our incorporation of IMU and DVL sensor fusion to improve odometry, and described custom model training for YOLOv8 for the target underwater objects. In the next chapter, we describe our main results, processing a BlueROV2 dataset obtained in the MIT Z-Center pool.

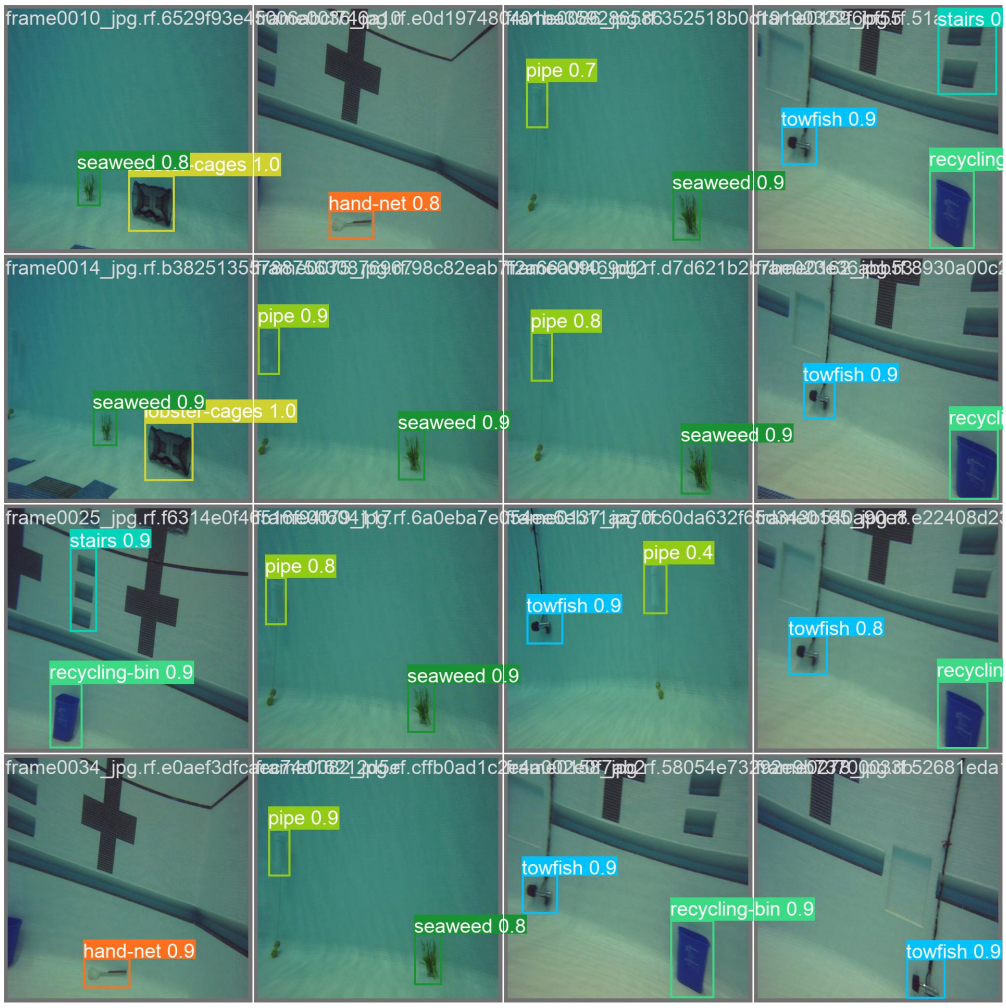


Figure 4.5: Batch of 16 generated test images for validating prediction in the target environment. Compared to the original input, generated images are randomly blurred and rotated.

Chapter 5

Results and Analysis

5.1 Introduction

This chapter provides the main results of the thesis: experimental evaluation of the GAPSLAM system adapted to operate underwater using the BlueROV2. We use the same experimental setup as used by Singh et al. [63] to evaluate their OAS-SLAM method. A series of objects were placed in the Z-Center pool near the pool walls. The main difference between their experiment and ours is in the selection of data, which results from the differing methodology. However it is convenient that we can make a direct comparison between the two methods.

GAPSLAM uses both Gaussian and NG landmarks as well as reinitialization to improve the back-end optimization of SLAM, while OAS-SLAM utilizes GA only to represent uncertainties in measurements and states, using the iSAM2 solver. OAS-SLAM instead focuses on object detection as well as trajectory and map accuracy. Precision and recall for both the PA and UMAP camera-sonar sensor fusion methods are compared against YOLO. This is a significant difference from underwater GAPSLAM, which focuses only on mapping and trajectory estimating using a monocular camera, with object classes known in advance and uses pre-trained detectors. Meanwhile, the novel PA and UMAP algorithms of the OAS-SLAM identify, localize, classify, and map marine objects without *a priori* knowledge of object classes by leveraging unsupervised object segmentation. This advantage is showcased in the OAS-SLAM outdoor tank dataset, which contains objects that cannot be detected by off-the-shelf object detectors in their default settings. Our approach works only for objects that fit one of the *a priori* known object classes. The metric these two SLAM systems have in common, and the metric to evaluate any SLAM system is the trajectory APE from a common experiment, which is the focus of this section.

5.2 Z-Center Pool Experiment

The BlueROV2 travels across the Z-Center indoor pool while maintaining a set altitude and an approximately three meter distance from the wall. The pool is twenty-five meters long, and the indoor pool dataset is comprised of three runs: the one object one loop run (*1obj1loop*) in UMAP only, the two objects one loop run (*2obj1loop*) in both PA and UMAP, and the two object two loops run (*2obj2loop*) in both PA and UMAP. During each loop, the

BlueROV2 is oriented to continuously face the same pool wall that is parallel to the path and travel in a line to the other side of the pool. The path was carefully steered to avoid the pool drain cover panels, which adversely effect the measurements of the DVL when the robot is directly above the panel. The loop is completed when the BlueROV2 retraces its path and returns to the location where it started from. Various underwater objects with at least two objects of each class (lobster cage, seaweed, hand net, towfish, tire, pipe) were evenly spaced and placed in a line parallel to the wall. A visualization of the experiment layout is given in figure 5.1.

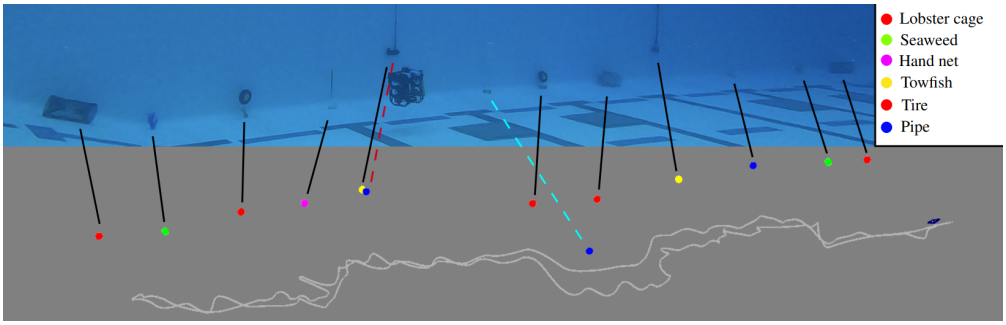


Figure 5.1: In order to compare YOLO GAPSLAM to the OAS-SLAM PA method [63], the same pool experiment dataset is used.

The following are the BlueROV2 sensors that are essential to GAPSLAM: odometry is obtained by fusing a Teledyne Impulse Doppler velocity log, a VectorNav IMU, and a Bar30 pressure sensor using an Extended Kalman Filter. A monocular Sony Exmor IMX322 camera with 1920 by 1080 resolution and 30 FPS is used to capture a stream of camera images [63].

Although a system for discerning GT was not available in the pool, setting up the experiment to guarantee loop closures, having a highly accurate odometry as well as having an accurate pseudo-GT comparable to the actual GT from a previous experiment provides a reliable framework for trajectory evaluation. Furthermore, OAS-SLAM and GAPSLAM are both in essence sparse landmark-based SLAM systems and therefore are more comparable to each other as they share distinct identifiable landmarks factors as reference points.

OAS-SLAM combined with YOLO was chosen as the pseudo-GT, which was also the pseudo-GT used by Singh et al. [63] for their indoor and outdoor experiments. The EKF odometry of BlueROV2 is highly accurate; in the outdoor experiment at WHOI [63] the APE of the odometry was only 0.02m more than PA OAS-SLAM, and surprisingly was 0.01m more accurate than pseudo-GT [63] when data from a low-cost Hybrid Long/Inverted Ultra-Short Baseline (iLBL-USBL) acoustic beacon system was used to obtain a GT trajectory. Therefore, when combined with loop closures, it was expected that the already quite reliable odometry would allow trajectories produced by OAS-SLAM to be even more accurate. YOLOv8, although not perfect in recall, has perfect precision across all indoor and outdoor experiments, and has the highest average score over PA and UMAP when precision and recall are combined. Although not perfect, this justifies selecting YOLO OAS-SLAM as pseudo-GT because it is expected to be highly accurate while being the best available method.

GAPSLAM was also compared to ground truth in the previous motion capture experi-

ment, and was demonstrated to be highly accurate. As such, it is expected for GAPSLAM, which uses the same reliable EKF odometry, to have a similar or better APE differential for the indoor pool experiment in comparison to the motion capture experiment.

5.3 Results

Tables 5.1 and 5.2, and 5.3 show some of our main results from the pool experiment, comparing our YOLO GAPSLAM against Singh et al.'s OAS-SLAM, for two different trajectories. There were mixed results. While OAS-SLAM performed significantly better than odometry, GAPSLAM only outperformed odometry in the *2obj1loop* run by 20 percent and the *2obj2loop* run by 10 percent. YOLO GAPSLAM did well considering that OAS-SLAM mitigates depth and scale ambiguity by making a direct measurement with the SONAR, but GAPSLAM did have the advantage of being able to use the same trained YOLOv8 as the pseudo-GT. As shown in figures 5.3, 5.5 and 5.7 the APE is focused in the z-axis just like in the Vicon room experiment.

Metric	**YOLO GAPSLAM**	**UMAP OAS-SLAM**	**EKF Odometry**
max	1.618	0.809	1.580
mean	0.943	0.079	0.817
median	0.973	0.057	0.847
min	0.183	0.007	0.098
rmse	0.979	0.119	0.866
sse	516.169	13.295	850.609
std	0.260	0.089	0.286

Table 5.1: APE w.r.t. translation part (m) with Sim(3) Umeyama alignment for YOLO GAPSLAM, EKF Odometry, and *1obj1loop* UMAP method. PA method was not available for this run.

Metric	**YOLO GAPSLAM**	**PA OAS-SLAM**	**EKF Odometry**
max	0.449	0.327	0.999
mean	0.195	0.108	0.233
median	0.191	0.113	0.106
min	0.007	0.009	0.006
rmse	0.225	0.126	0.344
sse	28.907	16.528	123.221
std	0.112	0.065	0.254

Table 5.2: APE w.r.t. translation part (m) with Sim(3) Umeyama alignment for YOLO GAPSLAM, PA OAS-SLAM, and EKF Odometry.

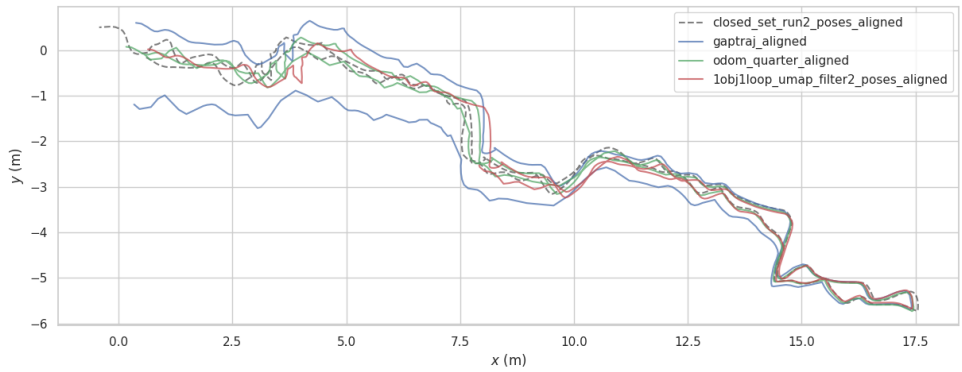


Figure 5.2: One object per each class, one loop of the pool. GT is in black, blue is GAPSLAM, green is EKF odometry, and red is UMAP OAS-SLAM.

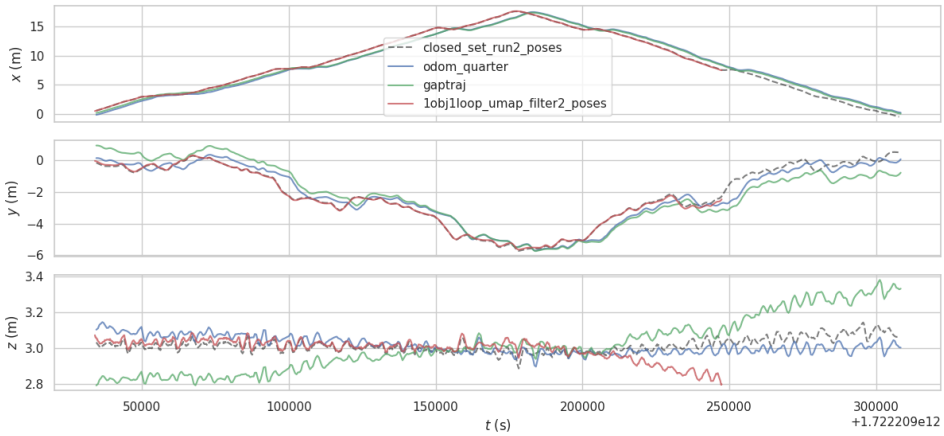


Figure 5.3: One object per each class, one loop of the pool in $x-y-z$ view. GT is in black, blue is GAPSLAM, green is EKF odometry, and red is UMAP OAS-SLAM.

Metric	**YOLO GAPSLAM**	**PA OAS-SLAM**	**EKF Odometry**
max	2.689	1.670	3.002
mean	1.288	0.970	1.409
median	1.251	0.930	1.300
min	0.292	0.260	0.113
rmse	1.343	1.020	1.513
sse	1933.778	109.550	3348.240
std	0.379	0.290	0.553

Table 5.3: APE w.r.t. translation part (m) with Sim(3) Umeyama alignment for YOLO GAPSLAM, *2obj2loop* PA, and EKF Odometry.

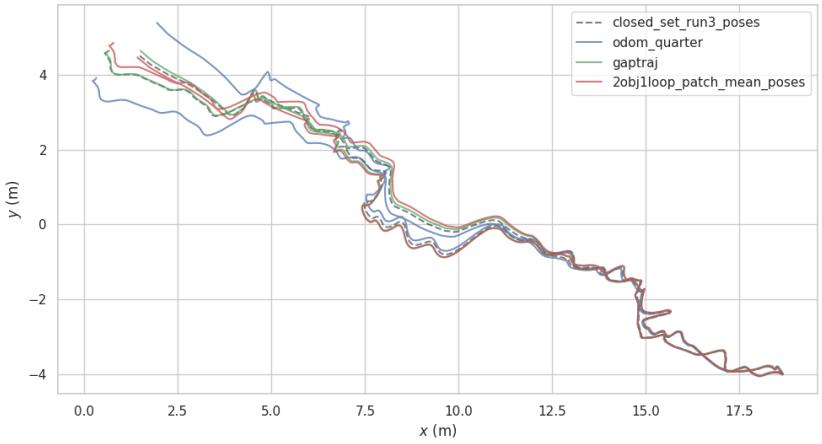


Figure 5.4: Two objects per each class, one loop of the pool. GT is in black, green is GAPSLAM, blue is EKF odometry, and red is PA OAS-SLAM.

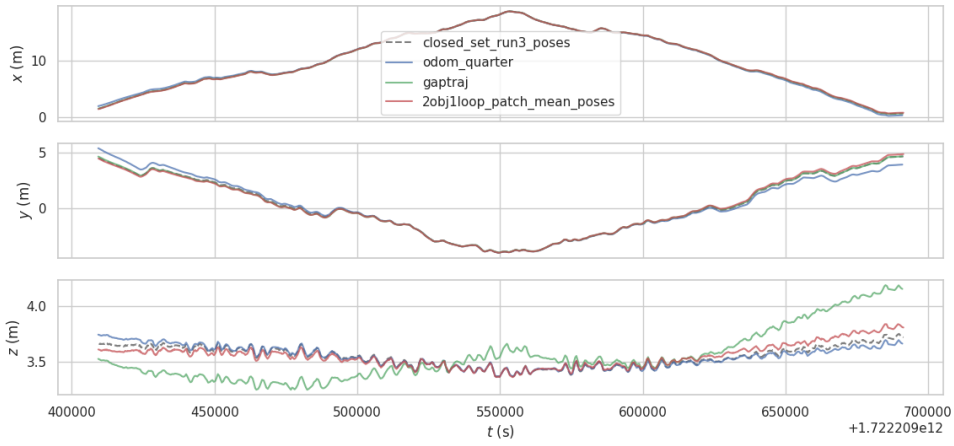


Figure 5.5: Two objects per each class, one loop of the pool in $x - y - z$ view. GT is in black, green is GAPSLAM, blue is EKF odometry, and red is PA OAS-SLAM.

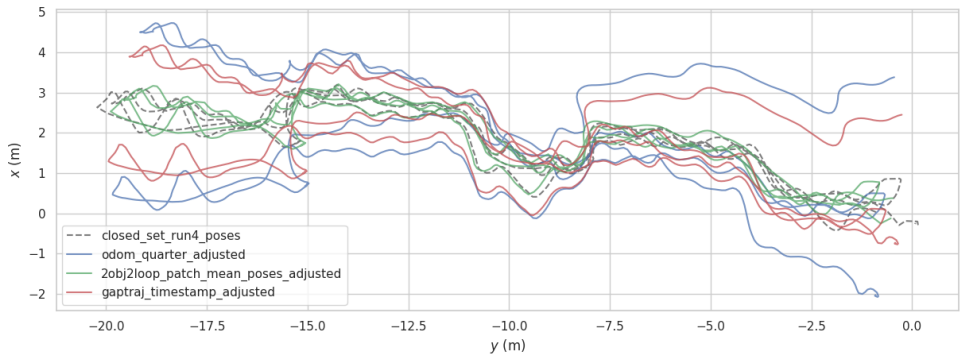


Figure 5.6: Two objects per each class, two loops of the pool. GT is in black, red is GAPSLAM, blue is EKF odometry, and green is PA OAS-SLAM.

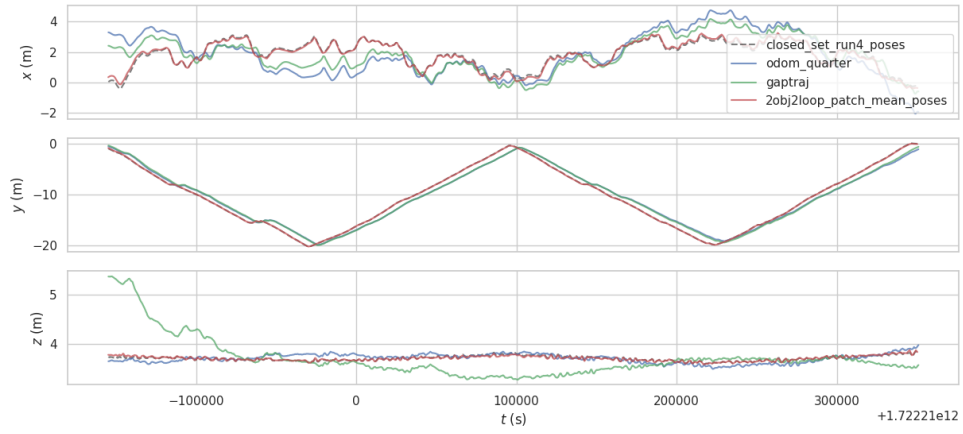


Figure 5.7: Two objects per each class, two loops of the pool in $x - y - z$ view. GT is in black, red is GAPSLAM, blue is EKF odometry, and green is PA OAS-SLAM.

5.4 Analysis

In our experiments, we encountered some shortcomings that require further study. Unless there is reinitialization of the GA of the landmark using linearization points from an associated NG particle filter via samples with the highest probability, and unless reinitialization reduces the largest eigenvalue of the samples of the marginal posterior of the landmark sufficiently, the system is still not confident about the landmark and it will remain in the NG set. Consequently the landmark, which is still not purely Gaussian, will not be recorded on the map for the final result as shown in figure 5.8. This poor quality map that consists of unvisualized particle filters is a problem for GAPSLAM because a lack of Gaussian-only landmarks not only indicates poor data association, but also a lack of confidence in the map's scale.

What was unexpected, however, was that GAPSLAM was able to perform relatively accurately for the *2obj1loop* run even though the system was having significant data association problems. For most of the scenario, the objects were constantly being detected as multiple distinct landmark observations of the same class, which added invalid NG factors to the pose graph. YOLO GAPSLAM underperformed in the *1obj1loop* experiment, which was somewhat expected as there were less objects to generate landmarks factors and complete loop closures. Additionally, its trajectory was not shorter than *2obj1loop* and it had equal potential to accumulate odometry errors. What was unexpected was that *1obj1loop* was outperformed by odometry, which indicates that while OAS-SLAM succeeded in completing its loop closure, GAPSLAM failed to make loop closures or made invalid ones such that the trajectory become adversely affected. One expected result that was observed was that the *2obj1loop* run performed significantly better than the *2obj2loop* run. Although there were more opportunities for loop closure in the *2obj2loop* run, travelling twice the distance was guaranteed to accumulate greater odometry drift, and ultimately affecting the SLAM trajectory as well.

In the indoor experiment by Huang [87], only 58 percent of runs without reinitialization returned solutions despite having RMSE values comparable to those with reinitialization. This record is consistent with our observations in the pool experiment. GAPSLAM without reinitialization is essentially the Gaussian solver with additional priors on landmarks. Underwater GAPSLAM had a reduced number of reinitializations due to poor data association and was not robust to errors encountered during MAP estimation. Changing the running parameters swayed the results excessively, which was not experienced in the Vicon room experiment. Effects of changing running parameters will be discussed in the next chapter. GTSAM 's indeterminant linear system detection error, which is the result of underconstrained variables and the system being underdetermined, occurred frequently if the running parameters were not set to optimal values. Furthermore, even for runs with identical running parameters, there was often a minor variance in the APE between each runs. Likewise, a previously stable configuration could end up unexpectedly not completing during a repeat run. Thus, our results were somewhat brittle, and more experimentation is required to take steps toward creating a robust underwater visual navigation system.

Figure 5.8: Map generated by YOLO GAPSLAM. Objects failed to reinitialize their Gaussian Approximations, due to incorrect data association. GAPSLAM does not record objects on the map unless they reinitialize and meet a confidence threshold.

5.5 Summary

In summary, this chapter has presented our main experimental results for underwater GAPSLAM in the Z-Center pool. While the results are preliminary, we achieved a basic proof-of-concept for underwater semantic SLAM. The next chapter further discusses some of the open questions arising from our work.

Chapter 6

Discussion

While previous chapter showed some success in providing the first implementation of GAPSLAM underwater, the experiments also highlight a number of difficult questions about data association and object detector performance in the underwater setting. In this chapter, we discuss a few of these issues and illustrate more of the challenges using data taken in a tank on the MIT campus, using Detic.

6.1 Limitations of Monocular SLAM

Monocular SLAM faces challenges during forward-facing sideways motion, including scale ambiguity, limited parallax, feature loss, and depth estimation difficulties. GAPSLAM was expected to mitigate some of these issues by using semantic landmarks for scale estimation and loop closures because Gaussian representations of object poses and particle filters help manage uncertainty for depth in sideways motion through incremental resampling and reinitialization. While semantic information and probabilistic techniques offer improvements over feature-based methods such as ORB-SLAM3, they don't resolve all issues inherent to sideways-only motion in monocular SLAM, as demonstrated by the relative reduction in trajectory accuracy going from the Vicon room to the Z-Center pool.

For monocular SLAM, the design of experiment is critical. OAS-SLAM while monocular, is resistant to depth or scale ambiguity because it can measure distance with sonar. GAPSLAM, which cannot leverage additional observation sensors is more susceptible to depth and scale ambiguity when there is a lack of diversity of views. Taking the same path to return to the starting point to finish the run may be advantageous for loop closures, but for monocular setups it is a compromise as the diversity of viewpoints is reduced. Additionally, in figures 3.2, 3.4 and 3.6 from chapter 3.3, it was seen that the majority of the APE originates from the z-axis, because a lack of motion in z equals a lack of opportunities for loop closures to affect z APE. Accurate monocular depth and scale estimation is crucial for constraining the z-axis. However, with motion only in the $x - y$ plane, depth estimation becomes more challenging, especially for features directly in front of the camera.

As such, the Vicon room experiment setup was essentially a $2D$ SLAM problem although the APE calculation in EVO was in 3D. Upwards and downwards translation and rotation, which occurred in the original GAPSLAM experiment should have been leveraged. However,

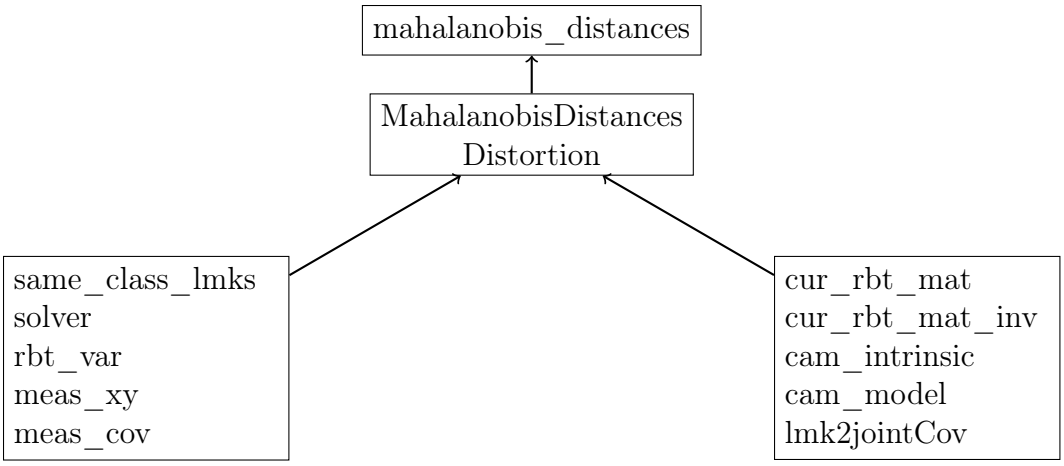


Figure 6.1: Mahalanobis Distances Distortion Function Diagram

there was a legitimate reason why z-axis motion was decided to not be included in the Vicon room experiments. Having a person carry the laptop connected to the Vicon system by ethernet, and connected to the D435i camera by USB was tricky, and attempts degraded experimental data due to bodies occluding the reflector balls from the Vicon cameras.

For the pool experiment, let $\hat{x}$ be the dimension that BlueROV2 primarily travelled during the experiment, let $\hat{y}$ be the dimension perpendicular to $\hat{x}$, going towards the objects and let $\hat{z}$ be the dimension parallel to the direction of gravity, where on one end is the air-water boundary and the pool bottom on the opposite end. Although not as lacking as motion in $\hat{z}$, the pool experiments had only minor translation in $\hat{y}$ compared to $\hat{x}$. As such, loop closures had the most effect on $\hat{x}$, which has the highest correlation with the x-axis, in all runs and it had the least amount of APE as shown in figure 5.3, 5.5 and 5.7. In hindsight, it was a blessing that BlueROV2 had to manoeuvre around the pool drains to avoid problems passing over the drains had on the DVL. Although this introduced a minor amount of motion blur, YOLO was robust to it and this added $\hat{y}$ motion and an increase in diverse viewpoints to allow GAPSLAM to have smaller APE than odometry for two of the three runs.

6.2 Mahalanobis Distances in GAPSLAM

Upon reviewing the logs generated during GAPSLAM runs, new NG objects were consistently created for the same object due to the larger than expected Mahalanobis distance between objects of the same class previously in chapter 2.3.4. Shown in figure 6.1, the *MahalanobisDistancsesDistortion* function outputs the Mahalanobis distance. If the Mahalanobis distance exceeds the threshold of 99 percent confidence level of the chi-squared distribution, then different-class objects are not rejected and same-class objects are not associated with each other, creating new landmarks. The threshold for the chi-squared distribution was incrementally increased and decreased, but changes in both directions had adverse effects, only increasing trajectory mean APE.

Mahalanobis distance in GAPSLAM is computed using several key parameters. The *same_class_lmks* represents the number of landmarks in a new frame that share a class

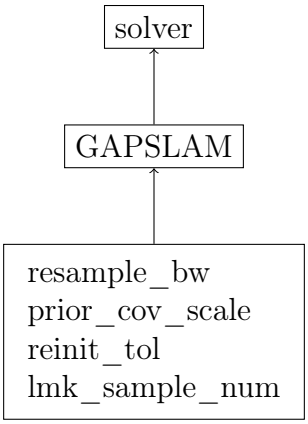


Figure 6.2: Solver Function Diagram

with an existing landmark and are within a specified *pixel_range* threshold. This pixel range is a crucial operating parameter in GAPSLAM. The calculation also incorporates the camera's intrinsic matrix and model, which are pre-calibrated using underwater checkerboard imagery.

The *solver*, an instance of the GAPSLAM solver class (detailed in Figure 6.2), is central to this process. It utilizes parameters such as resampling bandwidth for non-Gaussian landmarks, prior covariance scale, reinitialization tolerance for Gaussian landmarks, and the number of landmark samples. The solver's capabilities include retrieving joint covariance matrices for variables like the robot state variable (*rbt_var*).

Additional components include the *lmk2jointCov* dictionary, which stores covariance information for landmark-measurement associations, and the *cur_rbt_mat* and *cur_rbt_mat_inv*, which are $4x4$ matrices representing the robot's 3D pose used for coordinate frame transformations. The *meas_xy* and *meas_cov* parameters represent the mean coordinates and standard deviations of the object's center, derived from the object detector's segmentation mask.

As explained in chapter 4.5, the custom trained YOLOv8 model is only capable of detection, and not segmentation. Consequently, the entire bounding box of the object detection must be treated as the object segmentation mask. Normally, this would not be a problem as the center of the bounding box is typically the center of the object as well, or the discrepancy is negligible.

6.3 Troubleshooting

6.3.1 Unexpected Behavior of Custom YOLOv8 Model

The custom underwater YOLOv8 model had an unexpected behavior that would heavily affect the Mahalanobis distance calculation, which is crucial for data association. The bounding box of an object would consistently become smaller or larger between frames as shown in figure 6.3, even though there was negligible movement from the robot. To the GAPSLAM system, this would appear as the object becoming smaller and larger going from frame to

frame, without experiencing the necessary rotation or translation to warrant this perceived observation. Consequently, this could result in significant and unexpected fluctuations in the *meas_xy* and *meas_cov* parameters and wrongly push the resulting Mahalanobis distance outside of the threshold.

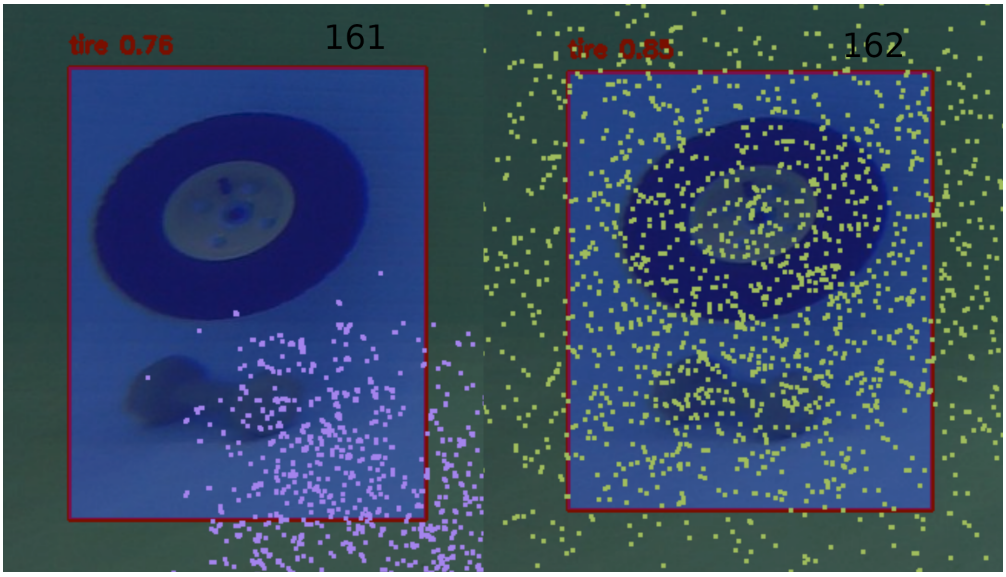


Figure 6.3: Frame 161 and 162 of the *2obj1loop* pool experiment run. The bounding box has thinner margins around the tire on the left side for 161.

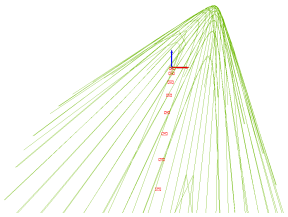
6.3.2 MIT Tank Experiment with Detic and GoPro9

In order to determine whether the inconsistent bounding boxes of the custom YOLOv8 detector was resulting in poor data association, another experiment was executed at the indoor tank of MIT Building 1-225, with object classes that had good precision and recall in Detic. The experiment used a GoPro9 camera containing an IMU. The camera and IMU were calibrated using the OpenIMUCameraCalibration method [109], and the video file was converted to a ROS bag file by the parsing method from Joshi et al. [110] previously used for underwater GoPro9 visual-inertial SLAM. The BlueROV2 was unsuitable for this experiment due to DVL errors experienced in the relatively shallow tank. Objects were aligned similarly to the Z-Center pool experiment, but the overall trajectory had a rectangular box shape instead of being near linear.

Figure 6.4: Initial frames of the tank experiment in MIT Building 1-225.

After the run was complete, it was realized that the tank experiment encountered the same problem as the pool experiment, which resulted in the runs failing to complete. The

changing shape of the bounding boxes was discovered to be only a minor part of the underlying problem. Seen in figure 6.4, the experiment starts at a good state where the particle filter is centered on the object detection, and going from frame to frame, the particles filter converges until reinitialization results in the removal of the NG landmark and displaying only the small green Gaussian-only landmark. Up until this point, this was identical to the successful GAPSLAM run in the Vicon room experiment.



(a) Unexpected increasing uncertainty in a Gaussian-only landmark after having its corresponding landmark removed removed from the NG set.

(b) New NG landmarks are growing more uncertain each frame.

Figure 6.5: Comparison of landmark uncertainties in different scenarios.

However, as seen in figure 6.5a, the Gaussian-only landmark that was expected to only become more accurate with additional views became extremely uncertain only after several frames from its removal from the NG set. Furthermore, as shown in Figure 6.5b, each new particle filter and their respective Gaussian landmark become more uncertain going from one frame to the next. From these observations, it can be inferred that there is error accumulating in the solver. This system behavior is not exclusive to the tank experiment; for certain configurations of the pool experiment, this error accumulation was observed as well, leading to GTSAM's indeterminant linear system detection error for underdetermined systems.

6.3.3 Effects of Running Parameters on GAPSLAM

Careful selection of a reduced *max_range*, increased *reinit_tol*, reduced *resample_bw*, reduced *samples_per_path* and reduced *min_unimodal_eig* allowed a stable Gaussian-only landmark to appear as shown in figure 6.7a, but this result had worse trajectory APE than the results from chapter 5.3, where there were no Gaussian-only landmarks.

During the pool experiment, it was discovered that these five from all the GAPSLAM running parameters show in figure 6.6 were the decision variables for underwater GAPSLAM. This is because increasing and decreasing the five parameters as described above adds artificial constraints to the system, limiting range and density of particle filter sampling as well as restricting Gaussian reinitialization and removal of particle filters. While this decreases the risk of encountering a GTSAM indeterminant linear system detection error while allowing loop closure to reinitialize a single Gaussian-only landmark, it had adversely effected the

remainder of the NG landmark observations, reducing overall trajectory APE. The best results from chapter 5.3 was achieved by balancing constraints from the five decision variables with better uncertainty handling, avoiding particle degeneracy, and properly exploring the state space for NG landmarks.

When considering the results from chapter 5.3 and figure 6.7, it can be concluded that whether the goal is to aim for better trajectory accuracy or aim for a more stable system, there is a limit on how well the system can perform due to the limits of monocular visual SLAM systems discussed previously, as well as due to the EKF odometry covariance currently being unknown, which is discussed in the next section.

6.3.4 Visual vs Inertial Odometry Covariance

From the results of chapter 6.3.2, it can be determined that Mahalanobis distance data association and the GAPSLAM solver, while important, are likely contributing negligibly to the observed errors. Even in the Vicon room experiments, multiple NG landmarks were created for some objects although ultimately there was no problem as data association executed correctly and only one landmark was reinitialized to Gaussian-only.

As seen at the beginning of the tank experiment, resampling, reinitialization and data association was operating as expected, but measurements became degraded due to error accumulation in the same manner as the pool experiment. The potential sources of this error include measurement noise, robot pose uncertainty, and odometry covariance. While measurement noise has been addressed via utilizing a more robust object detector in the tank experiment and robot pose uncertainty is modelled by the prior covariance matrix that only affects the initial robot pose, the odometry covariance values for a new underwater environment have never been verified, and the default values from the original experiment was used without determining their viability. The odometry covariance, while not directly visible in the GAPSLAM data association algorithm, plays a crucial role in propagating uncertainty through the system, affecting both Gaussian and NG landmark estimations.

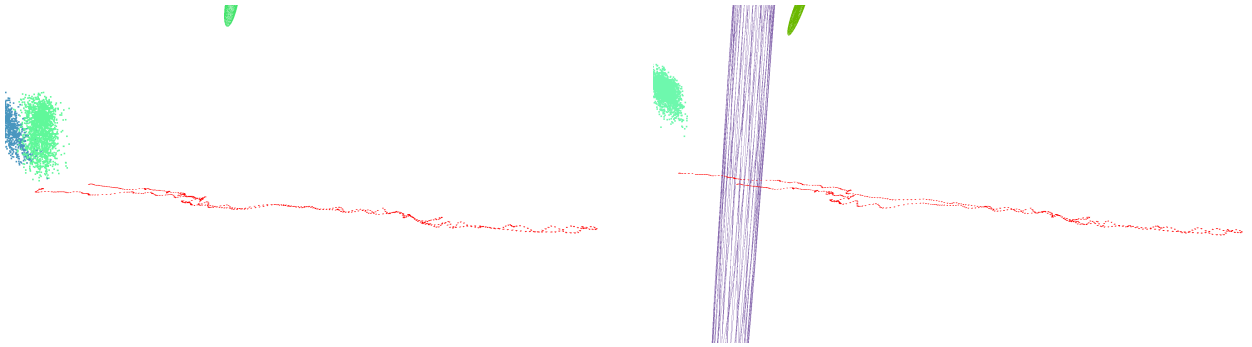
The odometry covariance is proportionally scaled by a critical running parameter called *translation_scale*. While this framework was sufficient for modelling uncertainty for GAPSLAM utilizing VO, it has been revealed by experimentation that this is not the case for inertial EKF odometry utilizing sensors from a real robot, where the errors are measured in metric units and not scalar values such as ratios or pixel dimensions. Odometry covariance values as well as *translation_scale* have been arbitrarily and incrementally increased and decreased, with adverse effects or no effect on the trajectory APE.

Further research is needed to better model the various sources of uncertainty discussed above, including underwater camera calibration, to better realize our dream of a robust object-based SLAM system for the challenging underwater domain.

Listing 6.1: YAML Configuration

```yaml
artificial_prior_cov: 1
detection_score: 0.5
min_da_pts_prct: 0.1
min_max_range:
- 0.1
- 20.0
min_mhl_dist: 9.21034037197618
min_unimodal_eig: 0.1
odom_pose_cov:
- 6.853891945200942e-06
- 6.853891945200942e-06
- 6.853891945200942e-06
- 9.0e-06
- 9.0e-06
- 9.0e-06
preprocess: true
prior_pose_cov:
- 0.0001
- 0.0001
- 0.0001
- 0.0001
- 0.0001
- 0.0001
rbt_path_num: 1
reinit: true
reinit_tol: 0.03
resample_bw: 0.02
rot_tol: 1.7453292519943294e-06
sampleDA: true
sample_per_path: 500
scale: 1
trans_tol: 3.0e-05
```

Figure 6.6: YAML Configuration for the system running parameters.



(a) GAPSLAM map from run with reduced *max_range*, increased *reinit_tol*, reduced *resample_bw*, reduced *samples_per_path* and reduced *min_unimodal_eig*, which reduces the effectiveness of NG landmarks but results in one successful Gaussian-only landmark during loop closure for one of the earlier landmarks.

(b) GAPSLAM map from a run with a configuration similar to the map on the left. However, when a second Gaussian-only landmark is reinitialized, the Gaussian mean and joint covariance continues to grow rapidly. Fortunately, this run completed before the GTSAM indeterminant linear system detection error triggered.

Figure 6.7: The green Gaussian-only landmark has been partially cut out from the visualizations in both the left and right GAPSLAM maps because the extrinsic matrix that transforms the map's observer from the robot camera's intrinsic matrix was not expected to require reconfiguration while in actuality it did. This is due to an unexpected and significant variance in the trajectory that occasionally occurs from one run to another.

Chapter 7

Conclusion

7.1 Summary of Contributions

In summary, our main contributions have been as follows:

1. This work develops the design of a visual semantic SLAM system capable of running on underwater robots such as AUVs and ROVs.

2. This work explores ground truth trajectory evaluations of SLAM systems in order to validate their accuracy.

3. This work establishes benchmarks for the proposed SLAM system by comparing it with other state-of-the-art underwater SLAM systems, aiming to identify areas for further improvement.

7.2 Comparison with Related Work

Compared to OAS-SLAM, the advantage of underwater YOLO GAPSLAM is that at 10 FPS it runs appropriately fast for real-time operation on medium to low-speed robots like BlueROV2. While OAS-SLAM processes data association in real-time in a similar manner to GAPSLAM, the open-set object detector using PA and UMAP is computationally demanding and consequently unable to operate at a real-time rate expected for BlueROV2, even with medium to high-end commercial graphical processors. However, our implementation of GAPSLAM is unable to operate without prior knowledge of the environment, and appears to be significantly less accurate in missions that lack rotation resulting in a lack of diverse viewing angles, leading to depth and scale ambiguity. Thus we can see the trade-off when attempting to create a real-time system, there is a drop in performance and capability.

The following summarizes lessons learned for GAPSLAM from the Z-Center pool and GoPro9 tank experiments discussed in chapter 6:

1. Object detectors that do not have consistent and tight bounding boxes around detections introduce additional measurement noise, which is not critical but may degrade GAPSLAM performance.

2. Key running parameters can loosen or tighten constraints for GTSAM, and manipulating them can balance the trade-off between performance and stability.

3. Both the pool and tank experiments start off as expected with particle filters converging at the center of object bounding boxes. However, as the run progresses new particle filters initiate with more uncertainty and are less likely to start converging. This is likely due to improperly modelled odometry covariance.

4. Experimental setup is critical for monocular systems like GAPSLAM. Diversity of views for objects and balanced movement in x, y, z as well as roll, yaw and pitch are critical to performance.

7.3 Future Research

Our work raises many questions for future research, primarily related to uncertainty modeling, domain shift, and data association. We discuss more in these topics below.

7.3.1 EKF Odometry Covariance Measurement

In order to progress with improving GAPSLAM performance on BlueROV2, proper knowledge of its odometry uncertainty is required. Measuring the EKF odometry covariance matrix for a real underwater robot involves a systematic process of data collection and analysis. To determine the values for each element of the covariance matrix, the robot would need to be operated in a controlled environment where its true position and orientation can be accurately tracked, possibly using external sensors or a motion capture system. Multiple trials of the robot's movement should be conducted, comparing the EKF's estimated position and orientation with the ground truth measurements. The variances (σ^2) for each dimension (x, y, z) and rotation (ϕ, θ, ψ) would be calculated from the differences between the estimated and true values across these trials. This process accounts for the uncertainties in the robot's motion model and sensor measurements. The resulting covariance matrix provides a quantitative measure of the uncertainty in the robot's odometry estimates, which is crucial for accurate state estimation and navigation in underwater environments. Other running parameters should then be tuned using the new odometry covariance matrix as a reference.

7.3.2 Domain Shift

An important improvement to the current configuration of underwater GAPSLAM would the refine the underwater YOLO model to include more underwater object classes and develop it to both detect and segment objects with more consistency in the bounding boxes and masks. In addition, it is important to consider that while fine-tuning YOLO from the baseline method performs well, it has received training data of the exact objects in the same scene and lighting conditions as the test data. This means that even if the experiment is replicated in the same pool in the future, if there are any visual changes at all from the original experiment there is no guarantee that the model trained will enjoy the same precision and recall it did in the datasets used in the experiment [63].

This is primarily due to domain shift, which refers to the discrepancy that occurs when a machine learning model is trained on one domain (source domain) but tested on a different domain (target domain). This is particularly challenging in the context of underwater scenes due to the unique and variable conditions found underwater. Consistent results, while likely in this case, are not guaranteed. And of course, the target environments of interest for Naval applications are much more challenging.

Lighting is not the only impact on domain shift as water color, turbidity, marine life or a lack thereof, different cameras, and other environmental conditions can result in domain shift and cause object detectors trained in a different domain to not operate as expected. Furthermore, Chen et al. [111] have demonstrated that domains can be analysed and mapped against each other. Some domains overlap, allowing object detectors of one domain to operate correctly in the intersection zone of multiple domain distributions while others do not. In order for underwater GAPSLAM to operate robustly across various underwater settings, the challenge of domain-shift must be resolved in future research.

Image enhancement, used in previous works for feature-based visual SLAM is not a solution for underwater visual semantic SLAM. Wang et al. [112] discovered that underwater image enhancement suppresses the performance of object detection. Especially, it inhibits the detector to detect the hard cases as the image enhancement may introduce noise and increase the interference caused by the background. The current SOTA in domain-shift mitigation for universal underwater object detection is domain adaptation [113] [114] and domain generalization [115]. Both domain generalization and domain adaptation aim to address the challenge of domain shift but there are key differences in their approaches and assumptions:

Domain Adaptation

Domain adaptation involves training a model on source domain data while adapting it to perform well on a target domain. During training, access to the target domain data is available, which can be either labeled or unlabeled. The training process includes techniques to align the source and target domains, ensuring the model performs effectively across both. Key techniques for domain adaptation include feature alignment, where the feature distributions of the source and target domains are aligned using methods like Maximum Mean Discrepancy (MMD) [116] or adversarial training. Instance reweighing assigns different weights to source domain instances based on their similarity to target domain instances. Domain-adversarial training involves adversarial learning, where a domain classifier is trained to distinguish between source and target domain features, while the feature extractor is trained to confuse the domain classifier. Additionally, generating target-like data using techniques like UW-GAN that was discussed in the related works helps create target-like data from source domain data, further enhancing the model's adaptability.

Domain Generalization

Domain generalization aims to train a model that is inherently robust to domain shifts and can generalize well to unseen domains without having access to the target domain data during training. The model must be robust to domain shifts by learning from multiple

source domains, utilizing only source domain data during training, which emphasizes the need for robust feature extraction and generalization capabilities. Key techniques for domain generalization include feature alignment across source domains, ensuring that features of the same class from different source domains are similar, often using contrastive learning or semantic alignment losses. Data augmentation techniques increase the diversity of training data through methods like domain mixup, which interpolates between features from different domains, and style transfer, which transforms images from one domain to another. Self-supervised learning employs self-supervised tasks to improve the robustness of the model without requiring labeled data. Meta-learning [117] trains the model to quickly adapt to new domains by learning from a few examples. Aggregation-based methods use multiple domain-specific classifiers and aggregate their outputs to enhance generalization.

7.3.3 Batch Data Association

Data association is another aspect for improvement in any SLAM system. While the current heuristic for data association for GAPSLAM is tolerable given accurate odometry, good initialization, precise object detector and diverse viewpoints, a data association scheme that is more robust and less sensitive to a variety of system running parameters in adverse conditions like the pool experiment is desired. Zhang et al. [118] proposed K-means clustering for batch data association in his approach for Data-Association-Free SLAM (DAF-SLAM). K-means clustering is a technique for solving the SLAM problem where the associations between measurements and landmarks are unknown. The method involves both an inner and an outer optimization loop. The inner loop focuses on solving for the robot poses, landmark positions, and data association given the k number of landmarks.

$$\min_{x \in SE(d)^n, y \in \mathbb{R}^{d \times K}} \left(f_{odom}(x) + \sum_{k=1}^{m} \min_{j_k \in [K]} ||R_{i_k}^T (y_{j_k} - t_{i_k}) - \bar{z}_k||_{\Sigma}^2 \right) \tag{7.1}$$

Here, x represents the robot poses, y represents the landmark positions, $\bar{z}_k$ is the measurement of a landmark position, i_k is the index of the robot pose associated with the k-th measurement, j_k is the unknown index of the landmark, and R_{i_k} and t_{i_k} are the rotation and translation components of the robot pose x_{i_k}. Σ represents the measurement noise covariance matrix.

The DAF-SLAM algorithm solves this problem through an iterative process. It begins with an initial guess for the robot poses x. Each measurement is then projected to the world frame based on the current estimate of x: $\hat{z}_k = R_{i_k} \bar{z}_k + t_{i_k}$. Landmark positions y are initialized using k-means++ on the projected measurements: $y = \text{k-means++}(\hat{z}_1, \ldots, \hat{z}_m, K)$. Measurements are then assigned to landmarks using k-means clustering: $y, j_1, \ldots, j_m = \text{k-means}(\hat{z}_1, \ldots, \hat{z}_m, K)$. Finally, robot poses x and landmark positions y are updated given the current data associations j_k: $x, y = \text{SLAM}(x, y, j_1, \ldots, j_m)$. These steps are iterated for a fixed number of iterations (set to 15 in the paper).

The outer problem focuses on estimating the number of landmarks K. This is formulated as:

$$\min_{K \in \mathbb{N}} (f_{slam}^*(K) + \beta K) \tag{7.2}$$

where $f^*_{slam}(K)$ is the minimized SLAM objective value for a given K, and β is a parameter that penalizes the use of more landmarks. The outer loop uses a multi-resolution gridding technique. It starts with the interval [1, m], where m is the number of measurements, and uniformly evaluates $f^*_{slam}(K) + \beta K$ at $N_K = 11$ values within this interval, including both endpoints. It then finds the optimal K_{n^*} out of these N_K values, shrinks the interval to $[K_{n^*-1}, K_{n^*+1}]$, and repeats until the interval size is 1. This approach makes approximately $N_K \log_{N_K/2}(m)$ evaluations of $f^*_{slam}(K)$, which is more efficient than naive gridding over all possible K values.

This combined approach leverages both efficient clustering methods and robust SLAM optimization techniques. While Gaussian-only landmarks in GAPSLAM are generally well associated, majority of the landmarks generated in GAPSLAM are NG, often generated from the same object. Under adverse conditions such as the one encountered in chapter 5 where only NG landmarks are available, K-Means clustering has potential for improving data association by reconnecting observation factors initially associated with the multiple additional NG landmarks that were incorrectly generated from the same object. Subsequently, combining K-Means clustering with the class names and number of objects provided by YOLO for each video frame, an avenue for pruning the non-existent landmarks can also be pursued. Applying the batch data association method described here to our underwater data would be an exciting topic for future research.

7.3.4 Interface with Control System via Mission Planner

Linear and non-linear Proportional–Integral–Derivative (PID) control systems have been designed for the BlueROV2 by Wu [119] via immersion tank testing, and simulated on MATLAB. In order for an autonomous mission planning module to control the BlueROV2, there must be an interface between the GAPSLAM map and t he mission planner, as well as an interface between the mission planner and the PID controller. The combined system would be able to generate a map and operate within without being given the map in prior. Rosen et al. [120] provides a block diagram architecture of such a system in figure 7.1.

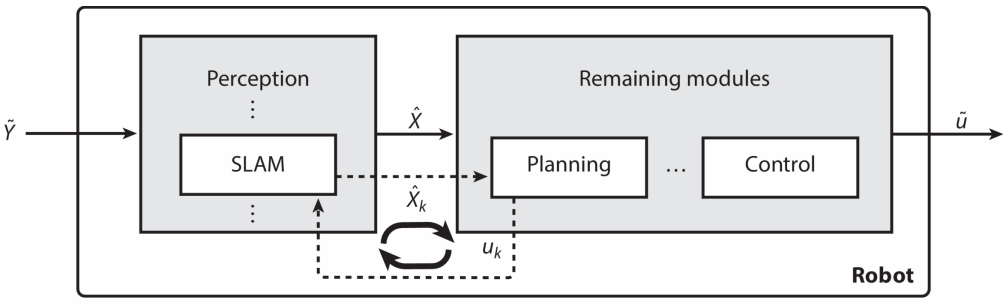


Figure 7.1: Sensor measurements $\tilde{Y}$ are received and processed, and a state estimate $\hat{X}$ is communicated to the rest of the system. The dotted loop indicates active perception, where the planner proactively chooses its controls u_k to reduce the uncertainty in the predicted state estimate $\hat{X}_k$ that is expected after u_k is applied.

7.4 Conclusion

This thesis has presented the design, implementation, and evaluation of an online underwater visual-inertial semantic SLAM system for the BlueROV2 platform. The system leverages the GAPSLAM algorithm, adapted for the unique challenges of the underwater environment.

The experimental results demonstrated that the adapted GAPSLAM system achieves comparable accuracy to the state-of-the-art in trajectory estimation, marginally outperforming pure EKF odometry with only Non-Gaussian landmarks. However, the system faced challenges in landmark reinitialization and mapping due to data association issues, highlighting areas for future improvement. Active SLAM, in which the AUV actively seeks out good objects for loop closures, could be an interesting avenue for future research to improve loop closing performance.

7.5 Summary

In conclusion, this thesis has demonstrated the potential of adapting advanced SLAM techniques like GAPSLAM for underwater environments. While challenges remain, particularly in uncertainty modeling, semantic mapping and data association, the system's performance in trajectory estimation shows promise for enhancing the autonomy of underwater vehicles. The underwater GAPSLAM system represents a significant step towards more capable and versatile underwater SLAM systems, potentially advancing the capabilities of autonomous underwater exploration and mapping across various applications including environmental monitoring, resource exploration, and underwater infrastructure inspection.

References

[1] Research and Markets, *Global $2.65 billion unmanned underwater vehicles market 2017-2021*, https://globenewswire.com/news-release/2017/03/10/934263/0/en/Global-2-65-Billion-Unmanned-Underwater-Vehicles-Market-2017-2021.html, Accessed: 20 June 2024, 2017.

[2] L. Levin, "Sustainability in deep water: The challenges of climate change, human pressures, and biodiversity conservation," *Oceanography*, vol. 32, pp. 170–180, Jun. 2019. DOI: 10.5670/oceanog.2019.224. URL: https://doi.org/10.5670/oceanog.2019.224.

[3] E. Cody, "Oil prices and their impact on the global economy," *The New York Times*, Jan. 2008. URL: https://www.nytimes.com/2008/01/11/business/worldbusiness/11iht-oil.1.9147825.html.

[4] C. Bücker, *World ocean review 3, marine resources – opportunities and risks*, 2014. URL: https://wedocs.unep.org/20.500.11822/9553.

[5] J. Bao, D. Li, X. Qiao, and T. Rauschenbach, "Integrated navigation for autonomous underwater vehicles in aquaculture: A review," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 139–151, 2020, ISSN: 2214-3173. DOI: https://doi.org/10.1016/j.inpa.2019.04.003. URL: https://www.sciencedirect.com/science/article/pii/S221431731930071X.

[6] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, *et al.*, "Autonomous underwater vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience," *Marine Geology*, vol. 352, pp. 451–468, 2014, 50th Anniversary Special Issue, ISSN: 0025-3227. DOI: https://doi.org/10.1016/j.margeo.2014.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0025322714000747.

[7] H. Niu, S. Adams, K. Lee, and N. Bose, "Applications of autonomous underwater vehicles in offshore petroleum industry environmental effects monitoring," *Journal of Canadian Petroleum Technology*, vol. 48, pp. 12–16, May 2009. DOI: 10.2118/09-05-12-GE.

[8] Lloyd's Register, *Marine safety report 2023*, Lloyd's Register Group Limited, London, UK, Accessed: 20 June 2024, 2023. URL: https://www.lrfoundation.org.uk/publications/marine-safety-report-2023.

[9] W. Zhang, P. Zhuang, H.-H. Sun, G. Li, S. Kwong, and C. Li, "Underwater image enhancement via minimal color loss and locally adaptive contrast enhancement," *IEEE Transactions on Image Processing*, vol. 31, pp. 3997–4010, 2022. DOI: 10.1109/TIP.2022.3177129.

[10] R. J. Davies-Colley and D. G. Smith, "Optically pure waters in Waikoropupu Springs, Nelson, New Zealand," *New Zealand Journal of Marine and Freshwater Research*, vol. 29, no. 2, pp. 251–256, 1995. DOI: 10.1080/00288330.1995.9516658.

[11] A. Jesus, C. Zito, C. Tortorici, E. Roura, and G. De Masi, "Underwater object classification and detection: First results and open challenges," in *OCEANS 2022 - Chennai*, IEEE, Feb. 2022. DOI: 10.1109/oceanschennai45887.2022.9775417. URL: http://dx.doi.org/10.1109/OCEANSChennai45887.2022.9775417.

[12] B. Joshi, N. Vitzilaios, I. Rekleitis, *et al.*, "Experimental comparison of open source visual-inertial-based state estimation algorithms in the underwater domain," Nov. 2019, pp. 7227–7233. DOI: 10.1109/IROS40897.2019.8968049.

[13] Parks Australia, *Geospatial data: Coral sea marine park - detailed boundaries*, https://atlas.parksaustralia.gov.au/node/50640, Accessed: 2024-08-08, 2024.

[14] X. Hua, X. Cui, X. Xu, S. Qiu, Y. Liang, X. Bao, and Z. Li, "Underwater object detection algorithm based on feature enhancement and progressive dynamic aggregation strategy," *Pattern Recognition*, vol. 139, p. 109 511, 2023, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2023.109511. URL: https://www.sciencedirect.com/science/article/pii/S003132032300211X.

[15] D. Park, W. Chung, and J. Kim, "Analysis of electromagnetic waves attenuation for underwater localization in structured environments," *International Journal of Control, Automation and Systems*, vol. 18, pp. 575–586, Mar. 2020. DOI: 10.1007/s12555-019-0548-9.

[16] I. I. Smolyaninov, Q. Balzano, and A. B. Kozyrev, "Surface electromagnetic waves at seawater-air and seawater-seafloor interfaces," *IEEE Open Journal of Antennas and Propagation*, vol. 4, pp. 51–59, 2023. DOI: 10.1109/OJAP.2022.3231885.

[17] M. H. G. Thomas, "The use of GPS for underwater navigation, sea trial results," in *Proceedings of the 8th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1995)*, Palm Springs, CA, Sep. 1995, pp. 949–955.

[18] J. Zhang, Y. Han, C. Zheng, and D. Sun, "Underwater target localization using long baseline positioning system," *Applied Acoustics*, vol. 111, pp. 129–134, 2016, ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2016.04.009. URL: https://www.sciencedirect.com/science/article/pii/S0003682X16300937.

[19] Y. Cong, C. Gu, T. Zhang, and Y. Gao, "Underwater robot sensing technology: A survey," *Fundamental Research*, vol. 1, no. 3, pp. 337–345, 2021, ISSN: 2667-3258. DOI: https://doi.org/10.1016/j.fmre.2021.03.002. URL: https://www.sciencedirect.com/science/article/pii/S2667325821000522.

[20] X. Wang, X. Fan, P. Shi, J. Ni, and Z. Zhou, "An overview of key SLAM technologies for underwater scenes," *Remote Sensing*, vol. 15, no. 10, p. 2496, 2023. DOI: 10.3390/rs15102496. URL: https://www.mdpi.com/2072-4292/15/10/2496.

[21] B. Singh, *REMUS AUVs*, LinkedIn, Accessed: 24 June 2024, 2024. URL: https://www.linkedin.com/pulse/remus-auvs-baljeet-singh/.

[22] Woods Hole Oceanographic Institution, *REMUS 100*, https://www2.whoi.edu/site/osl/vehicles/remus-100/, Ocean Systems Laboratory, 2023.

[23] J. Zhou, T. Yang, and W. Zhang, "Underwater vision enhancement technologies: A comprehensive review, challenges, and recent trends," *Applied Intelligence*, vol. 53, pp. 3594–3621, 2023, Accepted 11 May 2022, Published 01 June 2022, Issue Date February 2023. DOI: 10.1007/s10489-022-03767-y. URL: https://doi.org/10.1007/s10489-022-03767-y.

[24] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643.

[25] D. Reis, J. Kupec, J. Hong, and A. Daoudi, *Real-time flying object detection with YOLOv8*, 2024. arXiv: 2305.09972 [cs.CV]. URL: https://arxiv.org/abs/2305.09972.

[26] T. Lai, "A review on visual-SLAM: Advancements from geometric modelling to learning-based semantic scene understanding using multi-modal sensor fusion," *Sensors*, vol. 22, p. 7265, Sep. 2022. DOI: 10.3390/s22197265.

[27] Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1403–1410 vol.2. DOI: 10.1109/ICCV.2003.1238654.

[28] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual SLAM," *Expert Systems with Applications*, vol. 205, p. 117 734, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117734. URL: https://www.sciencedirect.com/science/article/pii/S0957417422010156.

[29] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, "A comprehensive survey of visual SLAM algorithms," *Robotics*, vol. 11, no. 1, 2022, ISSN: 2218-6581. DOI: 10.3390/robotics11010024. URL: https://www.mdpi.com/2218-6581/11/1/24.

[30] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, Invited paper, Lisbon, Portugal, Sep. 2006.

[31] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014. DOI: 10.1109/JOE.2013.2278891.

[32] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. URL: http://mcb111.org/w06/KollerFriedman.pdf.

[33] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, Sep. 2012. URL: https://repository.gatech.edu/server/api/core/bitstreams/b3606eb4-ce55-4c16-8495-767bd46f0351/content.

[34] S. Godio, A. Carrio, G. Guglieri, and F. Dovis, "Resolution and frequency effects on UAVs semi-direct visual-inertial odometry (SVO) for warehouse logistics," *Sensors*, vol. 22, Dec. 2022. DOI: 10.3390/s22249911.

[35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[36] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[38] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[39] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, INSTICC Press, 2009, pp. 331–340.

[40] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in Computer Vision*, M. A. Fischler and O. Firschein, Eds., San Francisco (CA): Morgan Kaufmann, 1987, pp. 726–740, ISBN: 978-0-08-051581-6. DOI: https://doi.org/10.1016/B978-0-08-051581-6.50070-2. URL: https://www.sciencedirect.com/science/article/pii/B9780080515816500702.

[41] D. Nistér, O. Naroditsky, and J. R. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004*, IEEE Computer Society, 2004, pp. 652–659, ISBN: 0-7695-2158-4. DOI: 10.1109/CVPR.2004.265. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2004.265.

[42] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part II: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012. DOI: 10.1109/MRA.2012.2182810.

[43] Y.-H. Lee, C. Zhu, G. Giorgi, and C. Guenther, "Stereo vision-based simultaneous localization and mapping with ranging aid," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018, pp. 404–409. DOI: 10.1109/PLANS.2018.8373407.

[44] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "Evaluation of RGB-D SLAM systems," *Journal of Field Robotics*, vol. 29, no. 6, pp. 846–864, 2012. DOI: 10.1109/ICRA.2012.6225199.

[45] Y. Yang and G. Huang, "Acoustic-inertial underwater navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4927–4933. DOI: 10.1109/ICRA.2017.7989571.

[46] S. Balasubramanian, A. Rajput, R. W. Hascaryo, C. Rastogi, and W. R. Norris, *Comparison of dynamic and kinematic model driven extended kalman filters (ekf) for the localization of autonomous underwater vehicles*, 2021. arXiv: 2105.12309 [cs.RO]. URL: https://arxiv.org/abs/2105.12309.

[47] S. T. Krauss and D. J. Stilwell, *Unscented kalman filtering on manifolds for AUV navigation – experimental results*, 2022. arXiv: 2210.06510 [cs.RO]. URL: https://arxiv.org/abs/2210.06510.

[48] M. Betke and Z. Wu, "An introduction to data association in computer vision," in *Data Association for Multi-Object Visual Tracking*. Cham: Springer International Publishing, 2017, pp. 1–6, ISBN: 978-3-031-01816-9. DOI: 10.1007/978-3-031-01816-9_1. URL: https://doi.org/10.1007/978-3-031-01816-9_1.

[49] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment–a modern synthesis," *Vision algorithms: theory and practice*, pp. 153–177, 1999.

[50] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[51] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*, ser. Lecture Notes in Mathematics, G. Watson, Ed., vol. 630, Springer Berlin Heidelberg, 1978, pp. 105–116.

[52] Georgia Tech, *Georgia Tech Smoothing and Mapping (GTSAM): A Library for Smoothing and Mapping*, 2021. URL: https://gtsam.org/.

[53] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641.

[54] M. Grimaldi, D. Nakath, M. She, and K. Köser, "Investigation of the challenges of underwater-visual-monocular-slam," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-1/W1-2023, pp. 1113–1121, Dec. 2023, ISSN: 2194-9050. DOI: 10.5194/isprs-annals-x-1-w1-2023-1113-2023. URL: http://dx.doi.org/10.5194/isprs-annals-X-1-W1-2023-1113-2023.

[55] Y. Song, J. Qian, R. Miao, W. Xue, R. Ying, and P. Liu, "HAUD: A high-accuracy underwater dataset for visual-inertial odometry," in *2021 IEEE Sensors*, 2021, pp. 1–4. DOI: 10.1109/SENSORS47087.2021.9639465.

[56] S. Rahman, A. Q. Li, and I. Rekleitis, "SVIn2: An underwater SLAM system using sonar, visual, inertial, and depth sensor," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019. DOI: 10.1109/ICRA.2019.8907638. URL: https://cs.paperswithcode.com/paper/an-underwater-slam-system-using-sonar-visual.

[57] T. Guerneve, K. Subr, and Y. Petillot, "Underwater 3D structures as semantic landmarks in SONAR mapping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 614–619. DOI: 10.1109/IROS.2017.8202215.

[58] M. Machado, P. Drews, P. Núñez, and S. Botelho, "Semantic mapping on underwater environment using sonar data," in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, 2016, pp. 245–250. DOI: 10.1109/LARS-SBR.2016.48.

[59] M. dos Santos, P. Drews, P. Núñez, and S. Botelho, "Object recognition and semantic mapping for underwater vehicles using sonar data," *Journal of Intelligent & Robotic Systems*, vol. 91, pp. 279–289, 2018.

[60] J. McConnell and B. Englot, *Predictive 3d sonar mapping of underwater environments via object-specific bayesian inference*, 2021. arXiv: 2104.03203 [cs.RO]. URL: https://arxiv.org/abs/2104.03203.

[61] G. Vallicrosa, K. Himri, P. Ridao, and N. Gracias, "Semantic mapping for autonomous subsea intervention," *Sensors*, vol. 21, no. 20, 2021, ISSN: 1424-8220. DOI: 10.3390/s21206740. URL: https://www.mdpi.com/1424-8220/21/20/6740.

[62] Ó. Álvarez-Tuñón, L. R. Marnet, L. Antal, M. Aubard, M. Costa, and Y. Brodskiy, *Subpipe: A submarine pipeline inspection dataset for segmentation and visual-inertial localization*, 2024. arXiv: 2401.17907 [cs.CV]. URL: https://arxiv.org/abs/2401.17907.

[63] K. Singh, J. Hong, N. R. Rypkema, and J. J. Leonard, *Opti-acoustic semantic slam with unknown objects in underwater environments*, 2024. arXiv: 2403.12837 [cs.RO]. URL: https://arxiv.org/abs/2403.12837.

[64] O. Powar, "A review: Underwater image enhancement using dark channel prior with gamma correction," *International Journal for Research in Applied Science and Engineering Technology*, vol. V, pp. 421–426, Mar. 2017. DOI: 10.22214/ijraset.2017.3077.

[65] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. Romeny, and J. B. Zimmerman, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[66] K. Köser, Y. Song, L. Petersen, E. Wenzlaff, and F. Woelk, *Robustly removing deep sea lighting effects for visual mapping of abyssal plains*, 2021. arXiv: 2110.00480 [cs.CV]. URL: https://arxiv.org/abs/2110.00480.

[67] R. Mur-Artal and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

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

[69] J. Engel, T. Schoeps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," vol. 8690, Sep. 2014, pp. 1–16, ISBN: 978-3-319-10604-5. DOI: 10.1007/978-3-319-10605-2_54.

[70] J. Bian, S. Wang, C. Li, and M. Li, "BAD-SLAM: Bundle adjusted direct RGB-D SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 10 540–10 546.

[71] K. M. Jatavallabhula, S. Saryazdi, G. Iyer, and L. Paull, *Gradslam: Automagically differentiable slam*, 2020. arXiv: 1910.10672 [cs.RO]. URL: https://arxiv.org/abs/1910.10672.

[72] H. Gupta and K. Mitra, *Unsupervised single image underwater depth estimation*, 2019. arXiv: 1905.10595 [cs.CV]. URL: https://arxiv.org/abs/1905.10595.

[73] B. Yu, J. Wu, and M. J. Islam, *UDepth: Fast monocular depth estimation for visually-guided underwater robots*, 2023. arXiv: 2209.12358 [cs.CV]. URL: https://arxiv.org/abs/2209.12358.

[74] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991. DOI: 10.1109/34.88573.

[75] C. Allen-Blanchette, S. Leonardos, and J. Gallier, "Motion interpolation in sim(3)," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 339–345. DOI: 10.1109/ICRA.2014.6906900.

[76] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, ISSN: 1941-0468. DOI: 10.1109/tro.2016.2624754. URL: http://dx.doi.org/10.1109/TRO.2016.2624754.

[77] T. Qin, P. Li, and S. Shen, *A general optimization-based framework for local odometry and mapping with multiple sensors*, 2019. arXiv: 1901.03642 [cs.RO]. URL: https://arxiv.org/abs/1901.03642.

[78] Stereolabs, "ZED: Stereo camera for 3d perception," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2016, pp. 1–2. URL: https://www.stereolabs.com/zed/.

[79] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. DOI: 10.1177/0278364914554813.

[80] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018. DOI: 10.1109/TRO.2018.2853729.

[81] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[82] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. DOI: 10.1177/0278364915620033.

[83] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th. John Wiley & Sons, 2012, pp. 64–66, ISBN: 9780471754954.

[84] J. Sattar, G. Dudek, A. Xu, J. Zacher, M. Pleau, and C. Prahacs, "The Aqua autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 40, no. 2, pp. 330–339, 2015. DOI: 10.1109/JOE.2014.2353475.

[85] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic SLAM using a monocular camera," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 1277–1284.

[86] X. Wang, R. Girdhar, S. X. Yu, and I. Misra, *Cut and learn for unsupervised object detection and instance segmentation*, 2023. arXiv: 2301.11320 [cs.CV]. URL: https://arxiv.org/abs/2301.11320.

[87] Q. Huang and J. J. Leonard, *GAPSLAM: Blending gaussian approximation and particle filters for real-time non-gaussian SLAM*, 2023. arXiv: 2303.14283.

[88] J. L. Puga, M. Krzywinski, and N. Altman, "Bayes' theorem," *Nature Methods*, vol. 12, pp. 277–278, Mar. 2015. URL: https://doi.org/10.1038/nmeth.3335.

[89] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, USA: Springer, 2006, ch. 4, p. 213.

[90] P. Torma, A. György, and C. Szepesvári, "A markov-chain monte carlo approach to simultaneous localization and mapping," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 852–859. URL: https://proceedings.mlr.press/v9/torma10a.html.

[91] Y. Dodge, "Mahalanobis distance," in *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 325–326, ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_240. URL: https://doi.org/10.1007/978-0-387-32833-1_240.

[92] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, Jul. 1900. DOI: 10.1080/14786440009463897. URL: https://doi.org/10.1080/14786440009463897.

[93] X. Zhou, C. Jia, V. Koltun, and P. Krähenbühl, *Detecting twenty-thousand classes using image-level supervision*, 2022. arXiv: 2201.02605 [cs.CV]. URL: https://arxiv.org/abs/2201.02605.

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[95] X. Xie, G. Cheng, J. Wang, X. Yao, and J. Han, *Oriented r-cnn for object detection*, 2021. arXiv: 2108.05699 [cs.CV]. URL: https://arxiv.org/abs/2108.05699.

[96] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[97] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kuemmerle, C. Dornhege, H. Kleinehagenbrock, and T. Schulz, "A comparison of SLAM algorithms based on a graph of relations," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009, pp. 2089–2095. URL: https://gki.informatik.uni-freiburg.de/papers/burgard_et_al_iros09.pdf.

[98] B. Liu and N. Paquin, *Viconmavlink: A software tool for indoor positioning using a motion capture system*, 2018. arXiv: 1811.11878 [cs.RO]. URL: https://arxiv.org/abs/1811.11878.

[99] M. Grupp, *EVO: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo, 2017.

[100] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

[101] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[102] Stanford Artificial Intelligence Laboratory et al., *Robotic operating system*, version ROS Melodic Morenia, May 23, 2018. URL: https://www.ros.org.

[103] A. Gupta, P. Dollár, and R. Girshick, *Lvis: A dataset for large vocabulary instance segmentation*, 2019. arXiv: 1908.03195 [cs.CV]. URL: https://arxiv.org/abs/1908.03195.

[104] M. Barisic, S. Kragelund, T. Masek, and Z. Vukić, "An online auv trajectory replanning software architecture based on the moos," Jan. 2009.

[105] Ultralytics, *Raspberry pi: Convert model to NCNN and run inference*, 2024. URL: https://docs.ultralytics.com/guides/raspberry-pi/#convert-model-to-ncnn-and-run-inference.

[106] M. Galdamez, J. Muntz, and M. Ferretti, *Bluerov2: An open-source platform for rapid development and validation of autonomous underwater vehicles*, 2020. arXiv: 2011.10488 [cs.RO]. URL: https://arxiv.org/abs/2011.10488.

[107] D. Eddelbuettel, *A brief introduction to redis*, 2022. arXiv: 2203.06559 [stat.CO]. URL: https://arxiv.org/abs/2203.06559.

[108] Y. Xiao, X. Ruan, X. Zhang, and P. Dong, "Monocular visual-inertial state estimation for micro aerial vehicles," *MATEC Web of Conferences*, vol. 139, p. 00 068, Jan. 2017. DOI: 10.1051/matecconf/201713900068.

[109] S. Urban, *OpenICC: An open imu and camera calibrator*, https://github.com/urbste/OpenImuCameraCalibrator.

[110] B. Joshi, M. Xanthidis, S. Rahman, and I. Rekleitis, "High definition, inexpensive, underwater mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1113–1121. DOI: 10.1109/ICRA46639.2022.9811695.

[111] X. Chen, Y. Lu, Z. Wu, J. Yu, and L. Wen, *Reveal of domain effect: How visual restoration contributes to object detection in aquatic scenes*, 2020. arXiv: 2003.01913 [cs.CV]. URL: https://arxiv.org/abs/2003.01913.

[112] Y. Wang, J. Guo, W. He, H. Gao, H. Yue, Z. Zhang, and C. Li, *Is underwater image enhancement all object detectors need?* 2023. arXiv: 2311.18814 [cs.CV]. URL: https://arxiv.org/abs/2311.18814.

[113] Z. Wang, L. Shen, M. Yu, K. Wang, Y. Lin, and M. Xu, *Domain adaptation for underwater image enhancement*, 2021. arXiv: 2108.09650 [cs.CV]. URL: https://arxiv.org/abs/2108.09650.

[114] J. Wen, J. Cui, B. Zhao, B. Han, X. Liu, Z. Gao, and B. M. Chen, *A real-time framework for domain-adaptive underwater object detection with image enhancement*, 2024. arXiv: 2403.19079 [cs.CV]. URL: https://arxiv.org/abs/2403.19079.

[115] Y. Chen, P. Song, H. Liu, L. Dai, X. Zhang, R. Ding, and S. Li, *Achieving domain generalization in underwater object detection by domain mixup and contrastive learning*, 2023. arXiv: 2104.02230 [cs.CV]. URL: https://arxiv.org/abs/2104.02230.

[116] F.-X. Briol, A. Barp, A. B. Duncan, and M. Girolami, *Statistical inference for generative models with maximum mean discrepancy*, 2019. arXiv: 1906.05944 [stat.ME]. URL: https://arxiv.org/abs/1906.05944.

[117] J. Vanschoren, *Meta-learning: A survey*, 2018. arXiv: 1810.03548 [cs.LG]. URL: https://arxiv.org/abs/1810.03548.

[118] Y. Zhang, O. A. Severinsen, J. J. Leonard, L. Carlone, and K. Khosoussi, *Data-association-free landmark-based slam*, 2023. arXiv: 2302.13264 [cs.RO]. URL: https://arxiv.org/abs/2302.13264.

[119] C.-J. Wu, "6-dof modelling and control of a remotely operated vehicle," Academic Supervisor: Assoc. Prof. Karl Sammut, Master of Engineering Thesis, Flinders University, Adelaide, Australia, Jul. 2018.

[120] D. M. Rosen, K. J. Doherty, A. T. Espinoza, and J. J. Leonard, *Advances in inference and representation for simultaneous localization and mapping*, 2021. arXiv: 2103.05041 [cs.RO]. URL: https://arxiv.org/abs/2103.05041.