Simultaneous Localization and Mapping in Underwater Robots

by

Franco Hidalgo Herencia
BEng, MEng

THE UNIVERSITY OF
WESTERN
AUSTRALIA

This thesis is presented for the degree of Doctor of Philosophy of The University of Western Australia

School of Electrical, Electronic and Computer Engineering

Supervisors: Prof. Dr. Thomas Bräunl (Coordinating)
Dr. Adrian Boeing

February 2019

Thesis Declaration

This thesis contains **published work and/or work prepared for publication, some of which has been co-authored.** The bibliographical details of the work and where it appears in the thesis are outlined below.

F. Hidalgo, J. Mendoza and F. Cuéllar, "ROV-based acquisition system for water quality measuring" *OCEANS 2015* - MTS/IEEE Washington, Washington, DC, 2015, pp. 1-5. doi: 10.23919/OCEANS.2015.7404435. (Chapter 2)
The estimated percentage contribution of the candidate is 80%.

F. Hidalgo and T. Bräunl, "Review of underwater SLAM techniques" 2015 *6th International Conference on Automation, Robotics and Applications (ICARA)*, Queenstown, 2015, pp. 306-311. doi: 10.1109/ICARA.2015.7081165. (Chapter 4)
The estimated percentage contribution of the candidate is 80%.

F. Hidalgo, C. Kahlefendt and T. Bräunl, "Monocular ORB-SLAM Application in Underwater Scenarios" *OCEANS 2018* - MTS/IEEE Kobe. Accepted and in press. (Chapter 6)
The estimated percentage contribution of the candidate is 80%.

F. Hidalgo, C. Kahlefendt and T. Bräunl, "Experimental Evaluation of Monocular ORB-SLAM2 in Underwater Environments," *Journal of Machine Vision and Applications*, Springer. Under review. (Chapter 6)
The estimated percentage contribution of the candidate is 80%.

F. Hidalgo and T. Bräunl, "Mobile Robotics Control Framework Applicable to Underwater Robots" *ROBOTICS*, MDPI. Under review (Chapter 3)
The estimated percentage contribution of the candidate is 80%.

F. Hidalgo and T. Bräunl, "Interest Point Detectors and Descriptors for Underwater Visual SLAM" Robotics and Autonomous Systems, Elsevier. Under review (Chapter 5)
The estimated percentage contribution of the candidate is 80%.

Abstract

Water covers more than 70% of the surface of our planet, and there are still areas that remain largely unexplored. Underwater engineering research offers scientist a variety of technologies including robots and specialized instrumentation to explore this environment. Marine robot development faces different challenges from its construction to its control and navigation due to the highly dynamic and harsh conditions of this scenario, limitations in communication, instrumentation, and energy. In this dissertation, we aim to extend the development of underwater robot technologies by investigating and implementing robotics vehicles and, applying and evaluating localization and mapping approaches towards autonomous navigation. This thesis is organized as a collection of research manuscripts based on articles already published or submitted to internationally refereed conferences and journals.

In this dissertation, we research two main challenges in underwater robots. *First*, we focus on the implementation of underwater robots for scientific studies. We present the implementation of a novel Remotely Operated Vehicle (ROV)-based acquisition system based on current underwater sensors for scientific studies. The design and preliminary tests of the data acquisition are presented. Then we propose a robot framework based on a novel low-level expansion board which applies to underwater robots. We upgrade two underwater robots based on the framework including a simulation environment and Robot Operating System (ROS) integration.

Second, we focus on Simultaneous Localization and Mapping (SLAM) algorithms and their application to underwater scenarios. We review three main SLAM approaches and use them over collected data from a simulation for comparison. Then, we center in visual SLAM, for which, we gathered and made public available a collection of datasets from different underwater locations in various illumination conditions. We evaluate the performance of feature detectors and descriptors in matching features over consecutive frames of the datasets. Finally, we apply a visual SLAM method based on Oriented FAST and Rotated BRIEF (ORB) features and graph optimization. We present the resulting maps and trajectories generated and evaluate the algorithm over the datasets. We also offer the proper conditions and the challenges for its application.

Acknowledgments

I would like to express my sincere gratitude to my coordinating supervisor Prof. Thomas Bräunl for the continuous support during my research. His guidance and continuos feedback helped me in all the time of my research and writing this thesis.

I thank the staff members of the School of Electrical, Electronic and Computer Engineering (EECE) for their help and support during my candidature, specially to Leila Easter and Natalie Jagals for their administrative assistance and Linda Barbour for helping me to improve my writing.

I am thankful to all the students who were part of the Underwater Robots team. Specially to Chris Kahlefendt, Tim Raphael and Hendra Wahyu, whose support has been fundamental to my research.

I am grateful to the "Programa Nacional de Becas y Crédito Educativo: Beca Presidente de la República 2014-I" of the Peruvian Government for sponsoring my studies and my stay in Australia. Specially to Milagros Céspedes for her prompt and efficient management.

I want to thank my fellow postgraduates students at the University of Western Australia (UWA), Kai Li Lim and Marcus Pham, for all the time shared in the laboratory and their friendship. I am grateful to all my friends that became a second family during this years, Manuel Cárdenas, Roberta Dayrell and Nestor Vazquez. And specially I am grateful to Martin Richard for being with me in every single step of this journey.

Finally, I owe a great deal to my family. I want to thank my parents Renée Herencia and Hernando Hidalgo, and my brothers Nando and Hugo for their support and being with me despite the physical distance.

Contents

List of Acronyms **xiii**

List of Figures **xix**

List of Tables **xxiii**

1 Introduction **1**

1.1 Contributions 5

1.2 Thesis Overview 6

1.2.1 Chapter 2: Underwater Robots for Scientific Research: Implementation of an ROV-based acquisition system 6

1.2.2 Chapter 3: Hardware and Software Framework for Mobile Robots Applicable to Underwater Robots 7

1.2.3 Chapter 4: Underwater Robot SLAM Frameworks 8

1.2.4 Chapter 5: Interest Point Detectors and Descriptors for Underwater Visual SLAM 8

1.2.5 Chapter 6: ORB-SLAM Application in Underwater Environments 9

2 Underwater Robots for Scientific Research: Implementation of an ROV-based acquisition system **11**

2.1 Introduction 11

2.2 Water Resources Surveys in Peru and Methods 13

2.2.1 Fishing Gear 13

2.2.2 Oceanography and Water Quality 14

2.2.3 Underwater Noise 15

2.3 Vehicle Description 15

2.3.1 Interface and Control 17

2.3.2 Multiparameter Probe 20

2.3.3 Hydrophones 20

2.4 Preliminary Tests 22

2.4.1 Indoor Tests 22

2.4.2 Field Test 24

2.5 Chapter Summary 25

3 Hardware and Software Framework for Mobile Robots 27

3.1 Introduction 27

3.1.1 Related Work 28

3.2 Eyebot7 IO Board Structure and Motion Controller 30

3.2.1 Hardware Overview 31

3.2.2 Motion Controller 31

3.3 Interface: Command Set, Application Programming Interface (API) and ROS . 36

3.3.1 Command Set 36

3.3.2 API 37

3.3.3 ROS Integration 37

3.4 Simulator 38

3.5 Experiments 39

3.5.1 PID Controlled Wheel 41

3.5.2 Driving Test 41

3.6 Underwater Robots Implementation 43

3.6.1 MAKO 43

3.6.2 USAL 44

3.6.3 ROS Integration 44

3.7 Chapter Summary 46

4 Underwater Robot SLAM Frameworks 49

4.1 Introduction 49

4.1.1 Formulation of the SLAM Problem 50

4.1.2 The SLAM Process 51

4.2 Instrumentation for Underwater SLAM 52

4.2.1 Onboard Navigation Sensors 52

4.2.2 Exteroceptive Sensors 55

4.3 SLAM Frameworks for Underwater Environments 57

4.3.1 Extended Kalman Filter SLAM 57

4.3.2 FastSLAM 59

4.3.3 Graph SLAM / Smoothing . . . 62
4.3.4 SLAM in Underwater Applications . . . 63
4.4 SLAM Implementation in Underwater Environments . . . 65
4.4.1 SLAM Implementation . . . 65
4.4.2 Results . . . 68
4.5 Chapter Summary . . . 71

5 Interest Point Detectors and Descriptors for Underwater Visual SLAM 73
5.1 Introduction . . . 73
5.1.1 Related Work . . . 75
5.1.2 Feature Detectors in Visual SLAM . . . 75
5.1.3 Feature Detectors Evaluation . . . 76
5.2 Selected Feature Detectors and Descriptors . . . 76
5.3 Underwater Monocular Images . . . 78
5.3.1 Underwater Image Enhancement . . . 79
5.4 Evaluation Framework . . . 80
5.4.1 Detectable Features in Underwater Images . . . 81
5.4.2 Frame Sequence Matching . . . 81
5.4.3 Datasets . . . 81
5.4.4 Experimental Setup . . . 83
5.5 Results and Discussion . . . 84
5.5.1 Detectable Features in Underwater Images . . . 84
5.5.2 Frame Sequence Matching . . . 85
5.5.3 Image Enhancement . . . 87
5.5.4 Processing Time . . . 88
5.6 Chapter Summary . . . 90

6 Experimental Evaluation of Monocular ORB-SLAM2 in Underwater Environments 91
6.1 Introduction . . . 91
6.2 Related Work . . . 92
6.3 ORB-SLAM2 and the Underwater Scenario . . . 93
6.3.1 Algorithm Description . . . 93
6.3.2 ORB-SLAM2 Validation . . . 95
6.3.3 Underwater Challenges for Visual SLAM Approaches . . . 95
6.4 Evaluation . . . 96

6.4.1 Dataset Contents . . . 96
6.4.2 Evaluation Criteria . . . 97
6.4.3 Experiments . . . 99
6.5 Results . . . 107
6.5.1 Monotony . . . 107
6.5.2 Turbidity . . . 108
6.5.3 Dynamics . . . 108
6.5.4 Lighting . . . 109
6.6 Chapter Summary . . . 109

7 Conclusions 119
7.1 Summary . . . 119
7.2 Findings . . . 120
7.3 Future Work and Open Problems . . . 120

Appendix A 145

Appendix B 155

List of Acronyms

ARM Articulated Robot Motion

AKAZE Accelerated-KAZE

API Application Programming Interface

AUV Autonomous Underwater Vehicle

BA Bundle Adjustment

BRIEF Binary Robust Independent Elementary Features

BRISK Binary Robust Invariant Scalable Keypoints

C-KLAM Constrained Keyframe-based Localization and Mapping

CNN Convolutional Neural Network

COP Closed-form Online Pose-chain

CPU Central Processing Unit

CTD Conductivity—Temperature—Depth profiler

DLT Direct Linear Transformation

DoF Degrees of Freedom

DP Distributed Particle

DPPTAM Dense Piecewise Planar Tracking and Mapping

DSO Dense Sparse Odometry

DT Deferred Triangulation

DTAM Dense Tracking and Mapping

DVL Doppler Velocity Log

DVO Dense Visual Odometry

EIF Extended Information Filter

EKF Extended Kalman Filter

ESC Electronic Speed Controller

FAB-MAP Fast Appearance-based Mapping

FAST Features from Accelerated Segment Test

FPS Frames Per Second

GPS Global Positioning System

GPU Graphics Processing Unit

IMARPE Instituto del Mar del Perú

IMU Inertial Measurement Unit

IO Input-Output

JSON JavaScript Object Notation

KF Kalman Filter

LAN Local Area Network

LSD Large Scale Direct

MR Multi Robot

NID Normalized Information Distance

NN Nearest Neighbour

OKVIS Open Keyframe-based Visual Inertial SLAM

ORB Oriented FAST and Rotated BRIEF

PEM Photometric Error Minimization

PF Particle Filter

PTAM Parallel Tracking and Mapping

RANSAC Random Sample Consensus

RD Robust Dynamic

REBVO Realtime Edge-based Visual Odometry

REMODE Regularized Monocular Depth Estimation

RFM Relative Feature Measurements

RGB-D Red, Green, Blue and Depth

RK Robust Keyframe-based

ROS Robot Operating System

ROV Remotely Operated Vehicle

ROVIO Robust Visual Inertial Odometry

RPi Raspberry Pi

SCP Secure Copy Protocol

SEIF Sparse Extended Information Filter

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SPLAM Simultaneous Planning, Locating and Mapping

SURF Speeded-Up Robust Features

SVD Singular Value Decomposition

SVO Semidirect Visual Odometry

TCP Transmission Control Protocol

UAV Unmanned Aerial Vehicle

UDP User Datagram Protocol

UI User Interface

UKF Unscented Kalman Filter

UML Unified Modeling Language

US Ultra Sonic

UWA University of Western Australia

VIN Visual Inertial Navigation

VO Visual Odometry

vSLAM visual SLAM

WLAN Wireless Local Area Network

WMR Wheeled Mobile Robot

YAML YAML Ain't Markup Language

PID Proportional Integral and Derivative

DOF Degrees of Freedom

LED Light-Emitting Diode

NTU Nephelometric Turbidity Units

PSS Practical Salinity Scale

PWM Pulse Width Modulation

GPIO General Purpose Input/Output

RISC Reduced Instruction Set Computing

GLUT OpenGL Utility Toolkit

LCD Liquid Cristal Display

PSD Position Sensitive Device

AHRS Attitude and Heading Reference System

ASEKF Augmented State EKF

ASKF Augmented State Kalman Filter

ESEIF Exactly Sparse Extended Information Filter

FLS Forward Looking Sonar

ICP Iterative Closest Point

INS Inertiail Navigation System

MEMS Micro-Electro-Mechanical Systems

MES Multibeam Echo Sounder

MSE Mean Squared Error

MSIS Mechanically Scanned Imaging Sonar

RLG Ring Laser Gyro

SSS Side Scan Sonar

VAN Viewpoint Augmented Navigation

KLT Kanade-Lucas Tracker

FREAK Fast Retina Keypoint

GLOH Gradient Location and Orientation Histogram

CUDA Compute Unified Device Architecture

CSV Comma Separated Values

RMSE Root Mean Square Error

SERNANP Servicio Nacional de Áreas Naturales Protegidas por el Estado

UART Universal Asynchronous Receiver-Transmitter

SPI Serial Peripheral Interface

CAN Controller Area Network

List of Figures

1.1 Small and medium size ROVs . . . 2

2.1 Ghost fishing video footage – Ancon,Peru Courtesy of IMARPE . . . 14
2.2 ROV Assembly parts . . . 17
2.3 ROV components . . . 18
2.4 ROV exploded view – components . . . 19
2.5 Connection diagram of the ROV . . . 19
2.6 ROV interface . . . 20
2.7 ROV assembled . . . 20
2.8 Block diagram of hydrophone amplifier . . . 22
2.9 Multiparameter probe test . . . 23
2.10 Hydrophone array test . . . 23
2.11 Experiment execution . . . 24
2.12 Operation screenshots . . . 25

3.1 Eyebot7 IO Overview . . . 32
3.2 Differential drive/thrust robots . . . 33
3.3 $v - w$ Controller Diagram . . . 33
3.4 Identified model simulation and measured data from a real Wheeled Mobile Robot (WMR) car . . . 34
3.5 Simulation of the $v - w$ controller . . . 35
3.6 Velocity profile applied to v and w . . . 35
3.7 Simplified code for the velocity section of the $v - w$ controller . . . 36
3.8 Eyebot7 IO board interface . . . 37
3.9 MAKO simulation on EyeSim VR . . . 39
3.10 Trace of path followed by the MAKO (green line) . . . 40
3.11 MAKO simulation plot . . . 40
3.12 Speed Proportional Integral and Derivative (PID) Controller . . . 41

3.13 v-w square path test. 12 tests running subsequent commands to draw a square . 42
3.14 Error histograms for $v - w$ controller test 42
3.15 $v - w$ single loop analysis 43
3.16 USAL electronic connection diagram using an Eyebot7 IO and a RPi 45
3.17 Example of ROS implementation for underwater robots 47

4.1 SLAM process: (1) Make Initial map z_0. (2) Estimate location given u_k. (3) Make 2nd map z_1 from X_1. (4) Update location given z_1 (5) Update map based on: Durrant-Whyte et al. [84] 51
4.2 Block Diagram of the SLAM process Redraw from [53] 52
4.3 Underwater Acoustic Positioning Systems from: Paull et.al. [85] 53
4.4 Acoustic sensor for perception: Range Sonar, Image Sonar. Redraw from [105] . 55
4.5 Subsequent readings for image sensors. (a) Forward Looking Sonar (FLS) present overlapping areas between readings; (b) Side Scan Sonar (SSS) do not present overlapping between readings 56
4.6 Loop closing in SLAM 59
4.7 SLAM Graphical model Redrawn from [122] 60
4.8 Particle Filter simplified block diagram 62
4.9 GraphSLAM model. Each node is a pose. Edges represent observation constraints from landmarks and how x_j is seen from x_i Redrawn with permission from [89] . 63
4.10 Overall block diagram for a GraphSLAM system 63
4.11 Simulation Setup. (a) and (b) Underwater robot diving in a pool with fixed landmarks. (c) Processed images with data association, each color represents a different tagged landmark. 69
4.12 Estimated path and landmarks without SLAM, lines associate real and estimated landmark positions 70
4.13 Results of SLAM: path and landmarks 70
4.14 Position Mean Squared Error (MSE) at every step 71
4.15 Landmark MSE at the final step 72

5.1 Simplified vSLAM architecture 75
5.2 Lighting effects on underwater images 79
5.3 Sketch of the modified data flow. Blue arrows represents camera images, red arrows are IMU/barometer data, green arrows symbolize user input and orange arrows are Xsens accelerometer data. 82
5.4 Block diagram of data extraction for evaluation 84
5.5 Features extracted per dataset 85

5.6 Detail of features extracted from Dataset_1. b-f, frame 710 with features extracted, g and h show low features found 86
5.7 Detail of features extracted from Dataset_8. b-e, extractors application in different frames 86
5.8 Inliers, obtained after NN matches and homography, per dataset 87
5.9 Inliers ratios per dataset 88
5.10 Results for pre-processed Datasets 3, 4, 6 and 8 89
5.11 Processing time based on Dataset_2 89

6.1 ORB-SLAM2 Overview 94
6.2 Examples of challenges for Visual SLAM 98
6.3 Bar plot elements 98
6.4 ORB-SLAM validation test in a recreational pool 99
6.5 Pool dataset: (a) sample image. (b) example of poor feature distribution 100
6.6 UWA Pool selected bar graphs of ORB-SLAM2 states 100
6.7 ORB-SLAM2 results for pool experiment. (a) without, (b) with markers 101
6.8 Point Walter experiment: sample images 102
6.9 Point Walter experiment results 103
6.10 ORB-SLAM2 trajectory estimates for Point Walter experiments 104
6.11 Fremantle Marina experiment results 105
6.12 ORB-SLAM2 results for Fremantle Marina experiments 111
6.13 Fremantle Marina experiment sample images 112
6.14 Omeo wreck experiment results 112
6.15 Images showing problematic feature detection on areas consisting mainly of sand. While ORB-SLAM2 can find a lot of features in structured areas it is barely able to find any on sandy ground. 113
6.16 ORB-SLAM2 results for Omeo Wreck experiment 113
6.17 Boat experiment sample images 114
6.18 Boat experiment results 114
6.19 ORB-SLAM2 results for Boat experiment 115
6.20 A comparison of extracted features in monotonous areas with low (a) and high texture (b). The red dots resemble extracted corners 115
6.21 A frame showing algae covered rocks taken from dataset *exploring_slope*. (a) shows the extracted features in red. (b) illustrates matches found between the last frame's and the current frame's features in green. 116
6.22 Initialization in a highly dynamic environment 116

6.23 Tracking in a scene with moving algae 117
6.24 Feature extraction on two consecutive images of dataset *Rectangle* from Point Walter 117

1 Dataset 1 157
2 Dataset 2 157
3 Dataset 3 158
4 Dataset 4 158
5 Dataset 5 159
6 Dataset 6 159
7 Dataset 7 160
8 Dataset 8 160
9 Dataset 9 161
10 Dataset 10 161
11 Dataset 11 162
12 Dataset 12 162

List of Tables

1.1 Overview of Navigator ROV from TMT [1] 2

1.2 Overview of Remus 100 from Konsberg [5] 3

1.3 Overview of Girona 500 Autonomous Underwater Vehicle (AUV) from the Universitat de Girona [6] 3

2.1 ROV overview 16

2.2 Multi-parameter probe sensors 21

2.3 Hydrophone and amplifier parameters 21

3.1 Expansion boards for robotics applications comparison 29

3.2 Example of ASCII binary, API and ROS commands 38

3.3 Overview of BlueROV2 from Blue Robotics [82] 44

3.4 MAKO AUV: Technical specifications overview 45

3.5 USAL AUV: Technical specifications overview 46

4.1 SLAM approaches overview 64

4.2 Summary of selected works on underwater SLAM 66

5.1 Detectors / Descriptors characteristics and parameters 77

5.2 Profile parameters changes 80

5.3 Datasets characteristics 83

Chapter 1

Introduction

Underwater technology development is gaining importance in the scientific community as it allows researchers to perform surveys in oceans, seas, and lakes that were previously extremely challenging or impossible to achieve. The development of underwater robots presents challenges in the construction, communication, localization, control and deployment due to the nature of the environment. From a hardware perspective, various underwater robots have been proposed with a variety of architectures and sensors. However, there are few complete frameworks that allow a smooth integration of sensors and offer a simulated environment for testing before deployment. On the other hand, localization for underwater robots is needed for geo-referencing measurements and for navigation. Localization is a complicated and expensive task (e.g. acoustic positioning systems). Sensor fusion and Simultaneous Localization and Mapping (SLAM) methods offer alternatives to accomplish this purpose. However, there is room for exploiting these methods in the case of underwater scenarios since newer approaches have mostly been applied to land and air environments.

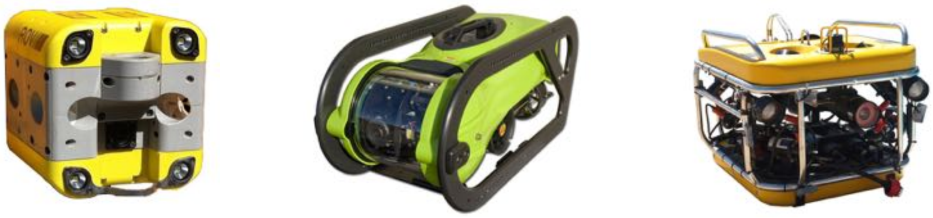
There has been a significant development in Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) in the last couple of decades. ROVs are wired to a station on the surface and controlled by an operator. They can manage powerful tools and provide high bandwidth for communication due to the tethered connection. For instance, the Navigator (Table 1.1) is a heavy-duty vehicle with four Degrees of Freedom (DOF) that can be equipped with hydraulic actuators and stream data from six cameras and other imaging sensors. Additionally, ROVs can have aided navigation systems such as depth control and orientation. They are used in inspection, maintenance, repair and structure deployment for the oil and gas industries.

Smaller ROVs are mostly used for visual inspection of dams, pipes, tunnels and structures. They are also used in habitat surveys, water sampling and quality measurements, and in fish

Table 1.1: Overview of Navigator ROV from TMT [1]

Overview	
Weight	1400 Kg
Size	1.2 m x 0.95 m x 1.2 m (L x W x H)
DOF	4 (surge, heave, sway, yaw)
Max Depth	600 m
Power	Electric / hydraulic
Actuators	
Thrusters	4 vectored, 1 vertical
Lights	6 available
Manipulator	Robot arm optional
Hydraulic	Guide wire cutter, mini dredge, plate handling, guidewire latch systems, mega digger
Sensors	
Cameras	Possible 6 cameras (2 pan/tilt)
Navigation	Doppler velocity log, depth gauge, fluxgate compass
Others	Sonar, multiparameter probe, rig floor monitor, densitometer

inspection. The AC-ROV [2] is a mini-ROV capable of inspecting 20cm diameter pipes and can access tiny spaces. Similarly, the Seabotix VLBV [3] and the Ocean-Modules V8 SII [4] are a small and a medium size ROV, respectively, that offer a configurable payload including water quality and acoustic sensors. The Seabotix VLBV has a gripper for cutting or handling objects.



(a) AC-ROV from AC-CESS [2]

(b) Seabotix VLBV from Teledyne [3]

(c) Ocean-Modules V8 SII from Ocean-ModulesOcean-Modules [4]

Figure 1.1: Small and medium size ROVs

AUVs are autonomous robots which rely on batteries and sensors to follow a set of waypoints that are evaluated continuously when the robot is able to communicate with a central station. AUVs can be loaded with a variety of hydrographic sensors and sophisticated localization and communication systems such as the Remus 100 (Table 1.2). The Girona 500 (Table 1.3 is an

AUV developed initially as a research platform which is now being commercialized due to its reliability over the last decade. It can also include an electric robot arm as a manipulator.

Table 1.2: Overview of Remus 100 from Konsberg [5]

Overview	
Weight	32 kg
Size	0.19 m x 1.70 m (Diameter x L)
DOF	3 (surge, pitch, yaw)
Max Depth	100 m
Battery	1.5 Kwh Li-Ion (12hrs)
Actuators	
Thrusters	1 DC brushless motor (surge)
Servomotors	2 gear motor and servo pot (fins for pitch and yaw)
Sensors	
Communication	acoustic, satellite and Wi-Fi
Navigation	Long base line (acoustic positioning system), doppler assisted-dead reckoning, intertial navigation system, GPS
Others	Side scan sonar, water quality

Table 1.3: Overview of Girona 500 AUV from the Universitat de Girona [6]

Overview	
Weight	200 Kg
Size	1.5 m x 1 m x 1 m (L x W x H)
DOF	4 (surge, heave, pitch, yaw)
Max Depth	500 m
Battery	2.2 Kwh Li-Ion (>6hrs)
Actuators	
Thrusters	4 to 8 configurable
Manipulator	Electric robot arm optional
Sensors	
Heading	Attitude and heading reference system
Depth	Pressure gauge
Navigation	Ultra short base line (acoustic positioning system), doppler velocity log
Others	Optional profiler sonar, side scan sonar, video camera

Sensors for monitoring the environment include underwater probes for water quality (which measures variables such as temperature, salinity, depth, turbidity and chemical composition). As well as acoustic sensors used for bathymetry (underwater topography), objects detection, biomass estimation, rough and detailed imaging, and for measuring underwater noise. Cameras provide easy to interpret information and are used at close proximity to the target of interest utilizing artificial illumination when needed.

GPS can not be used in underwater environments. Instead, acoustic is used for communication and localization by implementing networks of transducers and receivers to triangulate position. Doppler velocity logs use acoustics to determine velocity in reference to the water. Dead-reckoning sensors such as Inertial Measurement Unit (IMU) are used to estimate the orientation and the position of an underwater robot. Depth is easily measured through pressure sensors exploiting the direct proportionality between hydrostatic pressure and depth in underwater environments.

Accurate localization is crucial for robot navigation. Although acoustic positioning systems offer a reliable option, it requires expensive instrumentation and preparation. Different approaches have been researched to improve the estimation of the pose (position and orientation) of a robot based on the fusion of different sensors. SLAM is an alternative to integrate data from various sources, as per the sensors mentioned above, and can estimate the robot's pose on a map which is also built simultaneously.

Computer vision and development of optimization algorithms allow a specific SLAM approach based only on video cameras (visual SLAM). Motion is estimated from different viewpoints of overlapping frames when a robot is moving. The map is represented as a mosaic of images or as a cloud of points.

In this dissertation, we present contributions to two main challenges in underwater robots. **First**, we focus on the hardware and construction of underwater robots for scientific surveys. We investigate current underwater sensors for scientific studies based on the compilation of different trials and propose a novel ROV-based acquisition system. Furthermore, we propose a development framework for control and simulation of mobile robots with Robot Operating System (ROS) integration and apply it in the upgrade of two underwater robots. The framework is based on a novel low-level interface board for sensors and actuators which connects to a higher control unit such as a small computer. **Second**, we center on SLAM algorithms applied to underwater scenarios. We investigate three main approaches and adapt them to work with simulated data for proof of concept and comparison. Then, we research feature extractors on underwater images under the scope of visual SLAM (vSLAM) as an alternative based on cameras that exploit the advances in computer vision. We present a new dataset including a variety of scenarios and perform an extensive study on collected datasets to characterize their response and performance. Finally, we focus on ORB-SLAM which is a modern vSLAM method based on Oriented FAST and Rotated BRIEF (ORB) features extracted from camera images to create a path of the robot movement and a cloud of points as a map. We evaluate the algorithm for the datasets collected and present the conditions and limitations for its application in these scenarios.

1.1 Contributions

Most of the contributions presented in this thesis have been done entirely by the candidate except for:

- Chapter 2: the candidate was the Principal Researcher for the ROV project who directed and oversaw the implementation and tests of the ROV. The proper implementation was performed by the research group.

- Chapter 3: the candidate was in charge of the evaluation and modification of the Eyebot7 Board. He adapted the low-level functions based on a different microcontroller program. The C interface was performed by a teammate in constant communication and evaluation with the candidate. He oversaw the ROS packages integration and performed the tests presented in the thesis.

- Chapter 6: The candidate participated in the set-up of the ROV and the programs for logging the data with timestamps from the ROV. A Master's student and the candidate collected the datasets, except from the Fremantle Marina area which was obtained only by the student and the pool with markers which the candidate collected by himself. The Master's student run the ORB-SLAM2 algorithm and evaluated the algorithm supervised by the candidate. Finally, the candidate analysed the findings into the two ORB-SLAM2 manuscripts presented in the Thesis Declaration.

The major contributions of this thesis are as follows:

- We designed an implemented a novel ROV integrating three main sensors: a video camera, a multi-parameter probe, and hydrophones, for scientific underwater monitoring. (Chapter 2, Published in Oceans'14)

- We present a framework for robot development applicable to underwater robots based on the development of a low-level electronic board that interfaces sensors and actuators to the main controller. It includes connectivity to ROS and a simulation environment which allows a smooth transition from simulation to the implementation. (Chapter 3, submitted to Robotics)

- We propose the application of traditional SLAM frameworks based on filtering and non-filtering approaches to underwater robots. We evaluate their validity in a simulated environment and compare them theoretically and experimentally. We also describe common sensors used for this task. (Chapter 4, ICARA'15)

- We propose to exploit interest point detectors for underwater images captured by robots for visual SLAM. We categorize the challenges and evaluate the performance of common detectors and descriptors in experiments towards their application in vSLAM. We also make the datasets collected from a variety of underwater scenarios and conditions public. (Chapter 5, submitted to RAS)
- We propose the application of a visual SLAM approach (ORB-SLAM) in underwater scenarios. Datasets are used to evaluate the algorithm's performance. Limitations and key elements to its improvement are also discussed. (Chapter 6, Oceans'18 and manuscript submitted to MVA)

1.2 Thesis Overview

This thesis presents a number of novel solutions and studies related to the implementation and the application of SLAM algorithms to underwater robots. This dissertation is arranged as a collection of research manuscripts based on published or submitted to internationally refereed conferences and journals. The first two chapters are related to the implementation of mobile robots, mainly focused on the electronics, data acquisition and control. The following three chapters deal with the application of SLAM algorithms to underwater robots from the SLAM principles able to fuse different sensors and models to feature-based vSLAM.

Each of the chapters are a self-contained unit of information with an introduction to the topic and concluding remarks. We provide a brief overview of each of the chapters that follow this introduction in the description below.

1.2.1 Chapter 2: Underwater Robots for Scientific Research: Implementation of an ROV-based acquisition system

This chapter addresses the instrumentation and the use of underwater robots for scientific research of the hydrosphere in shallow water. It begins with an overview of ROV applications and current scientific studies for environmental studies carried out in the field and instrumentation setups.

We present a variety of scientific underwater studies trials performed, especially by the Instituto del Mar del Perú (IMARPE), and identify the instrumentation used. We present the implementation of a novel ROV-based acquisition system for scientific research. The implementation is divided into two parts:

(a) From the vehicle itself, we propose a novel ROV with the capabilities of mounting different sensors and integrating different sources of information into logs.

(b) From the instrumentation perspective and based on the needs of the IMARPE, we consider a video camera, standard in ROVs; a multi-parameter probe with interchangeable sensors; and an array of three hydrophones, for the measurement and detection of underwater noise.

We implemented the robotic platform and performed preliminary field tests to demonstrate its capabilities.

1.2.2 Chapter 3: Hardware and Software Framework for Mobile Robots Applicable to Underwater Robots

This chapter proposes a novel framework for developing mobile robots applicable to underwater vehicles based on a low-level expansion board. Robotics systems rely on direct tasks such as the interfacing with sensors and actuators, control routines and quick response to critical events to a logical or physical control unit in a hierarchy control architecture.

The framework proposed integrates the low-level board, an Application Programming Interface (API), a package for ROS integration and a simulation environment for mobile robots including underwater robots. These elements are briefly described below:

(a) The low-level expansion board can handle DC-motors, servo-motors, distance sensors, analog sensors, and audio. It has built-in control loops based on Proportional Integral and Derivative (PID) controller and a serial communication port for communication with a higher-level unit.

(b) The API is a collection of C functions that abstract the functionalities of the low-level board to develop high-level applications.

(c) The ROS package integration creates an accessible node in the ROS architecture that abstracts the API set for the low-level interface.

(d) The simulator is built on top of the API and includes a physical abstraction layer to simulate the behavior of modeled robots.

The framework is tested in wheeled robots and the upgrade of two underwater robots based on the framework is presented.

1.2.3 Chapter 4: Underwater Robot SLAM Frameworks

This chapter focuses on the implementation of SLAM on underwater robots. Underwater scenarios present different conditions compared to land or air for SLAM applications. For instance, the use of GPS is only possible when an underwater robot is on the surface; therefore, it can not rely on it for estimating its position when submerged. A range of sensors have been used for underwater localization and mapping applications including specific sensors for these scenarios such as sonars, echo sounders, radars and acoustic localization systems.

We present a selection of traditional localization and mapping sensors for underwater SLAM applications as well as an overview of SLAM frameworks applied to this scenario. We describe three main frameworks theoretically and implement them for a simulated environment. We simulate a pool with bricks and a robot moving over them with a camera mounted on the ROV looking down. The camera is used for detecting the bricks as simple landmarks, and simulated displacement sensor is used as a localization sensor.

We build the observation and motion models based on the simulated sensors and implement the principles of the frameworks to process the data in order to estimate the position of the robot and the landmarks. The simulation shows increased accuracy of the robot's estimated pose and the landmarks positions when SLAM is applied.

1.2.4 Chapter 5: Interest Point Detectors and Descriptors for Underwater Visual SLAM

This chapter deals with the processing of underwater images towards vSLAM. Feature-based vSLAM relies on matching features between frames. Images taken in underwater scenarios present different alterations (e.g. blurriness, sun flickering, changes in color) that affect the performance of common feature detectors and descriptors.

We performed an in-depth analysis based on collected datasets in different environments and situations that provide favorable outcomes concerning the performance of the sets detector/descriptor that, to the best of our knowledge, have not been documented. The descriptors are evaluated on consecutive frames and in situations where the robot re-observes the same area to recreate vSLAM conditions.

The results characterize which elements in the underwater scenarios show extractable features and the robustness of the descriptors towards vSLAM. Additionally, datasets that show low performance for the detectors/extractors are processed through enhancement algorithms and compared.

1.2.5 Chapter 6: ORB-SLAM Application in Underwater Environments

The last chapter of this dissertation presents an application of ORB-SLAM to the datasets collected in the previous chapter, evaluates its performance, points out the scenarios where the algorithm does and does not performs adequately, and suggests possible enhancements.

ORB-SLAM is based on ORB features which showed one of the best responses regarding the numbers of inliers detected in consecutive frames and reduced computational time at processing showed in Chapter 5. It uses bundle adjustment for optimizing the map which is composed of a cloud of points.

Chapter 2

Underwater Robots for Scientific Research: Implementation of an ROV-based acquisition system

This chapter addresses the application of underwater robots in scientific research from the design and implementation of a ROV-based acquisition system. It is designed to extend the capabilities of water quality monitoring and fishing gear surveys of Peruvian governmental research institutes related to the analysis and preservation of Peruvian water resources, such as rivers, lakes, and oceans. The robotic platform presented in this work, integrates an underwater video camera, a multi-parameter probe for water quality analysis, which measures parameters such as conductivity, temperature, depth, dissolved oxygen, salinity, pH and turbidity; and an array of three hydrophones to measure underwater noise and its direction. The versatility of an underwater vehicle is exploited by centralizing data acquisition and logging in a single flexible platform. Preliminary results of the ROV under controlled conditions for data acquisition tests, field tests in the open sea for navigation and its application in a scallop farm at the Peruvian National Park of Paracas are presented and discussed in this work.

2.1 Introduction

Water is a vital delicate resource for humanity. Either directly, such as for drinking water, energy generation, irrigation; or indirectly through the species and ecosystems dependent upon it. It is crucial for the sustainability of the planet to maintain its water resources alongside natural phenomena and anthropogenic activities. Water pollution including litter [7], debris [8], chemicals [9] and noise [10] can negatively alter underwater environments; therefore, the development of mechanisms for water quality monitoring is a critical aspect in the conservation process. Recent advances in underwater parameters instrumentation [11] and robotics allow the development of marine acquisition systems based on robots [12].

Nowadays, the development of underwater vehicles and robots for oceanographic and environmental analysis has become widespread among academic, public and private institutions. These types of robotic platforms are able to perform long-term operations, acquiring geo-referenced data and navigation maneuverability [11, 13]. In this context, underwater vehicles, such as ROVs (Remotely Operated Vehicles), AUVs (Autonomous Underwater Vehicles) and gliders are included as part of the instruments and equipment for field tests and data recollection [12, 14, 15]. These surveys allow the evaluation of marine areas where operations would otherwise be difficult for humans [16] such as the harsh environments of rivers, lakes and oceans, or in studies where conditions are threatening, particularly where water is polluted with solid waste [17], chemical waste [18] and underwater noise [19, 20] from anthropogenic and industrial activities. Common underwater sensors to evaluate these types of scenarios include imaging sensors [21], multi-parameter probes [14], and acoustic sensors [11]. Imaging sensors, such as cameras and side scan sonars, are used for visual inspections and mapping [8]. Underwater probes contain an array of sensors for measuring physical and chemical properties to evaluate the quality of the water environment in terms of pollutants and conditions for marine life sustainability [22]. Finally, hydrophones measure underwater noise, which is usually generated by machinery, explosives and the use of airguns in seismic inspections, and affects mostly marine mammals [10].

Underwater video allows direct observation of a target, and it is used for inspection operations such as visualization of fishing activities, marine flora, solid wastes, and others [23, 24]. Chemical and physical parameters are usually measured by Conductivity—Temperature—Depth profilers (CTDs) probes in oceanographic surveys [25]. Additional parameters, such as dissolved oxygen and pH [26] are also measured by special probes. Other specific analyses are performed in a laboratory through the collection of water samples in bottles [27]. As for underwater noise, it is measured by deploying arrays of hydrophones [28]. The intensity in time and spectral characteristics are used to determine acoustic signals in order to model certain underwater phenomena[29]. Localization of the source can be obtained by triangulation methods, according to the hydrophones array [30].

We present the design, implementation and preliminary tests of a Remotely Operated Vehicle (ROV) for water quality measurement which integrates three types of measuring technologies: visual imaging, water quality parameters, and underwater noise. The imagery is obtained through a video camera, while the water quality parameters (conductivity, temperature, depth, dissolved oxygen, salinity, pH and turbidity) are measured with a configurable six channel multi-parameter probe. The underwater noise is measured with an array of three hydrophones. The designed ROV is intended to be a research platform for educational institutions and a measurement tool for the Peruvian governmental research and conservation institutes in water resources, such as the Instituto del Mar del Perú (IMARPE), and the Servicio Nacional de Áreas Naturales Protegidas por el Estado (SERNANP).

The robotic platform presented in this document has been designed according to the technical requirements of the institutions described above towards the integration of different methodologies. The preliminary results of the ROV's data acquisition system under controlled conditions, field tests in open sea for navigation, and its application in a scallop farm at the Peruvian National Park of Paracas are presented.

This chapter is organized as follows. In Section 2.2, a brief review of the current state of water resource surveys made in Peru is described. Section 2.3 presents the description of the vehicle, its design and implementation. In Section 2.4, the evaluation of the integrated system in a controlled environment and in open sea are presented. Finally, in Section 2.5, the conclusions and future work for the robotic platform are discussed.

2.2 Water Resources Surveys in Peru and Methods

In Peru, the IMARPE is the national governmental institution oriented to the scientific research and study of the Peruvian sea and its hydric resources to advise the central government on the topics of fishery and marine life conservation. Its scope covers fishing gears technologies, oceanography and climate change, aquaculture and, pelagic and demersal fish studies. This section focuses on the chores in which underwater vehicles can be suitable for operations such as fishing gear performance—monitoring through direct observation; oceanography and water quality—through data acquisition from sensors; and underwater noise used for environmental studies.

2.2.1 Fishing Gear

Fishery is one of the most important economic activities in Peru. The most important marine resource, which is the target of local research, is the Peruvian Anchoveta (Engraulis ringens), a part of the anchovy family. Anchoveta are often found at a depth of 50m up to 150m and are caught via surrounding nets [31]. Other essential fishery resources are the jurel (Trachurus murphyi) and the Peruvian Merluza (Merluccius gayi peruanus) which are mostly caught through trawls [32]. Hook lines and modified versions of them are also used in the

Peruvian sea, especially for capturing giant squid (Dosidicus gigas); some species of mollusks and crustaceans are also gathered by pots and traps [33]. Additionally, aquaculture of the Peruvian scallop along with shrimps and river fish is a growing industry in the country [34].

One of the goals of the fishery, besides being efficient and cost-effective, is to be selective of the target species—which is fundamental for the sustainability of this activity. Another factor of concern is the effect of the fishing gear upon the habitat of the captured species by the destruction of the sea bottom, contamination from discards and ghost-fishing (lost or abandoned fishing gear that continues to capture species) [32].

Surrounding nets, hook lines, trawls, pots, and traps are studied by the IMARPE to characterize their behavior and selectivity. The surveys involve temperature, depth, bathymetry, motion sensors and video, followed by biological dissection of the caught [33, 35].

Video cameras have been used in [36] in a survey to study the impact of ghost fishing (Fig. 2.1). Other studies using video cameras are presented in [7, 8, 37].

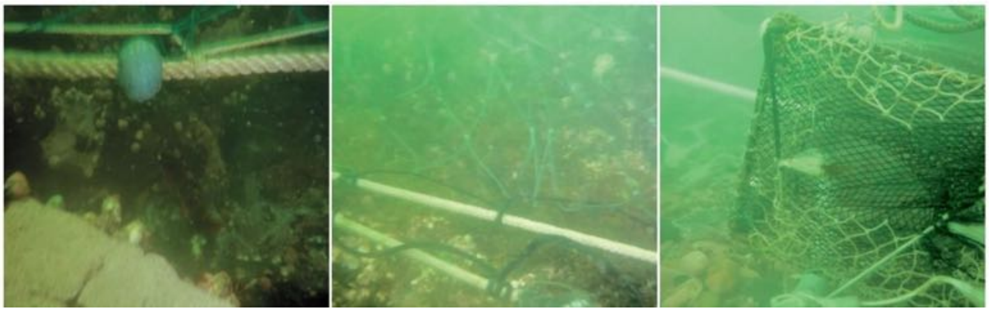


Figure 2.1: Ghost fishing video footage – Ancon,Peru
Courtesy of IMARPE

2.2.2 Oceanography and Water Quality

Physical and chemical properties of the water are part of the characterization of the environment of biological resources as well as human activities. Its findings are applied in fishing and climate change including the ‘El Niño’ phenomenon as well as water quality for sustainability and human health.

Sensor readings from research cruises are important for creating oceanographic maps [38]. Temperature, partially dissolved CO2, oxygen, salinity, chlorophyll, and pH are usually monitored to characterize a region or phenomena [39].

Depending on the intensity of ‘El Niño’, it can develop into heavy rainfalls, causing avalanches and mudslides as well as variations in water properties affecting the biodiversity [40]. To characterize its intensity, vertical temperature profiles, salinity and water samples are analyzed [41].

Water quality surveys are performed in different regions either for being ecosystems of interest or areas susceptible to contamination. The data is gathered from geo-referenced stations through CTDs and sample bottles for posterior laboratory analysis (salinity, pH and nutrients) [42]. Sometimes, a diver acquires the temperature of the bottom and collects samples of the sediment for organic material, carbonates, and pH analysis [27].

Around the world CTD probes are generally used to measure water parameters and their capabilities can be extended for large areas when mounted in small ships or gliders [14]. Besides cyclic phenomenon as El Niño, other recent arising phenomena such as global climate change, ocean acidification and littering are reflected in oceanographic parameters [43]. For instance, air pollution due to anthropogenic emissions of CO2 is exhibited in acidification of water resources presented as decreases in pH [26].

2.2.3 Underwater Noise

In general, underwater noise pollution comes from ships, underwater engines, explosives, and airguns. Any kind of energy produced in underwater scenarios affects the environment [43]. Marine species of fishes, cetacea, turtles, and their ecosystems can be adversely affected [19]. Airguns, used seismic prospecting, can emit intense acoustic waves in the search for resources in the oil and gas industry [10].

By measuring underwater noise, it is possible to register and measure the effects of activities that can affect the underwater environment. Shore constructions, oil station activities, boats, oil prospection and specific fishing gear can generate underwater noise in a variety of intensities and spread over long distances, affecting and altering marine wildlife species. For this reason, it is of particular interest for Peruvian institutions to recognize and measure noise generating activities such as the use of sounding devices in artisanal fisheries [44, 45], explosives in illegal fishing [29–31, 46], and seismic prospecting [47–49].

2.3 Vehicle Description

The robotic platform is designed as a ROV to work in the surveys described in Section 2.2. Therefore, a multiparameter probe and an array of hydrophones, besides a standard camera, are considered as the scientific instrumentation payload. The concept of the platform is to be flexible and to be able to log combined data.

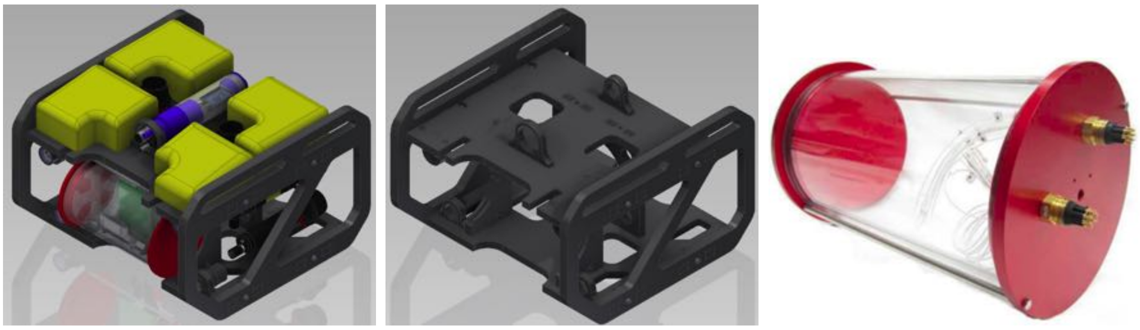
The ROV is connected to a computer on a vessel through a 120m tether. AC current and communication travels through the multicore tether. It allows a considerable bandwidth to stream large amounts of data, including video, sampled signals in real-time and control commands from an operator in the surface. The external structure is made of High-Density Polyethylene (HDPE) which is resistant to corrosion and makes it possible to work in fresh and sea water. It includes two handles at the top of the side panels () to simplify launching and recovering. The centerpiece is conformed by two parallel plates that support thrusters, buoys, sensors and two cylinders containing the electronic components. There are bolts on the top of the central plate to attach the buoys which compensate the buoyancy of the ROV. The dimensions are 0.75 (L) x 0.6 (W) x 0.55 (H)m with a weight in air of 50 Kg as seen in Table 2.1.

Table 2.1: ROV overview

Overview	
Weight	50 Kg
Size	0.75 m x 0.6 m x 0.55 m (L x W x H)
DOF	4 (surge, heave, yaw, roll)
Max Depth	100 m
Power	AC Electric
Actuators	
Thrusters	6 (4 horizontal, 2 vertical)
Lights	2
Sensors	
Cameras	1
Navigation	IMU, GPS, depth sensor
Others	Multiparameter probe, 3 hydrophones

The ROV has six 400HFS-L brushless thrusters, four mounted in horizontal position, and two, in a vertical position. The thrusters are placed symmetrically and the propellers are counter rotating to compensate the angular momentum (black and gold propellers in Fig. 2.3 (b)). This configuration provides five degrees of freedom to the ROV to be used: forward/backward, up/down, pitch, roll, and yaw. An exploded view of the ROV is shown in Fig. 2.4 where the position of the probe and the hydrophones can be observed.

Fig. 2.2 shows the 3D model of the ROV assembly (a), the external structure (b) and the main compartment for the electronics (c), which comes with underwater connectors. The electronic components are contained in two waterproof cylinders with underwater connectors for the devices mounted in the ROV structure Fig. 2.3 and Fig. 2.5. The multiparameter probe and the hydrophones can be mounted and dismounted easily from the ROV since they have independent connectors and mounting brackets. The main compartment contains the camera; a control board, based on a BeagleBone Black embedded computer and an expansion board to control the Light-Emitting Diode (LED) lights, and the hydrophones amplifier that connects to the controller analog channels. The secondary compartment contains the thruster drivers, power regulators for the electronics and the thrusters and a tether interface that allows interconnection of two computers over a wired Ethernet network above 120m. The detailed drawings of the ROV frame and a comprehensive connection diagram can be seen in Appendix A.



(a) ROV assembly model (b) ROV frame (c) Main compartment

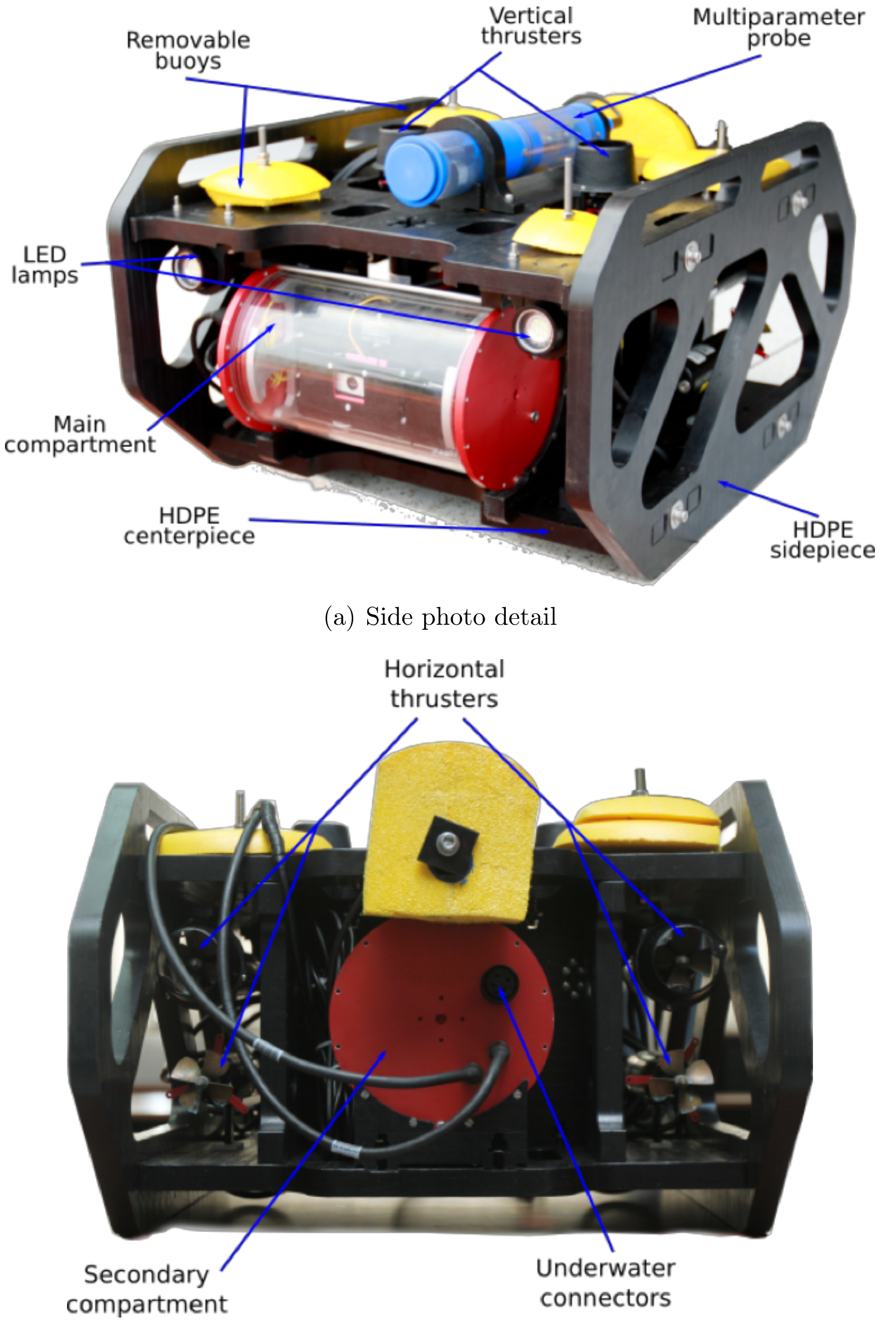
Figure 2.2: ROV Assembly parts

The BeagleBone Black is a general purpose Linux computer with a 1GHz Articulated Robot Motion (ARM) processor, which is also used as a data logger and communicates with an external computer, in the surface, for the robot's operation. The video stream, the probe data, and the signal from the hydrophones are stored along with the information of the ROV's heading and depth given by an IMU and the multiprobe's depth sensor.

2.3.1 Interface and Control

The embedded computer inside the ROV hosts a web server to monitor, control and download the logged data. The video is streamed online through an Ethernet connection to the external PC. The interface, which shows the streaming video along with the monitored parameters can be loaded from a web browser Fig. 2.6. The heading and depth are also displayed to provide information to the operator. Different bar indicators show the status of the power of the thrusters and the light intensity.

Regarding the ROV control, a gamepad is connected to the external computer. The analog sticks are used to proportionally control the power of the thrusters. The buttons allow to operate the LEDs and operate simple custom programs such as surfacing. The final integration of the robot can be seen in Fig. 2.7.



(a) Side photo detail

(b) Back photo detail

Figure 2.3: ROV components

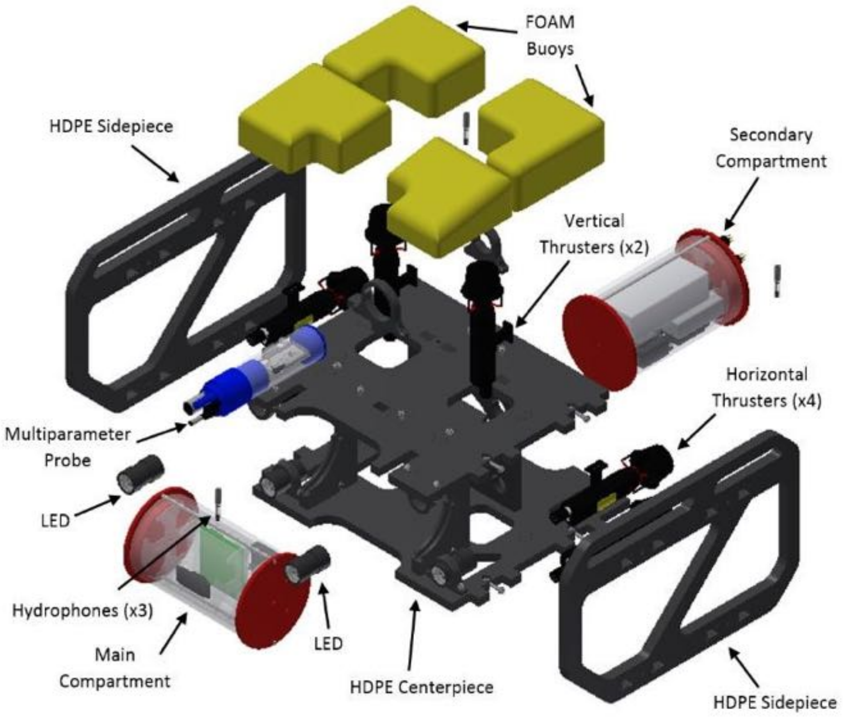


Figure 2.4: ROV exploded view – components

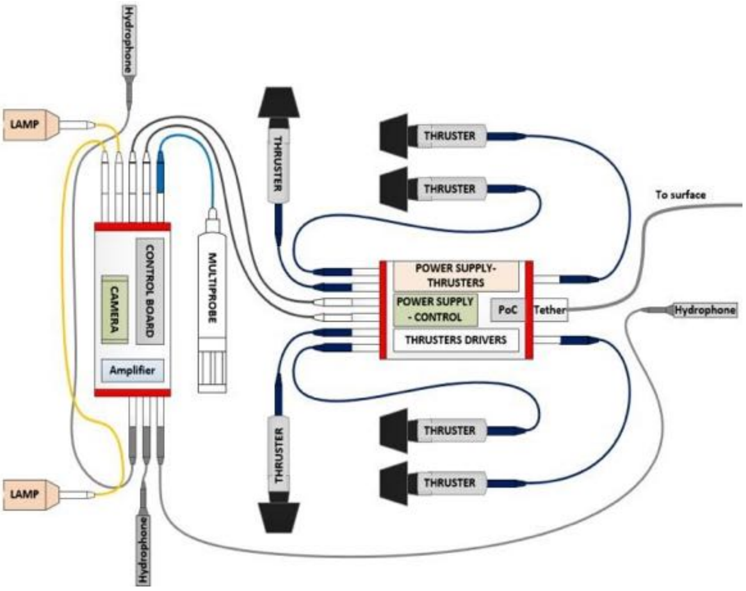
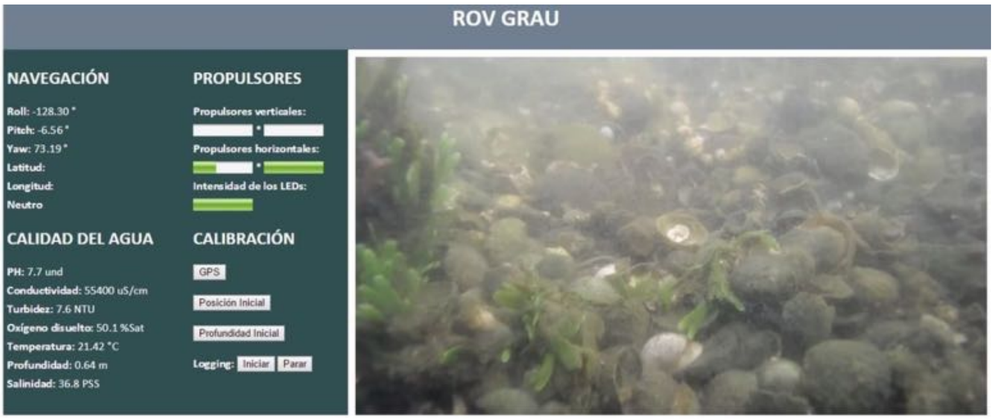


Figure 2.5: Connection diagram of the ROV



ROV GRAU

NAVEGACIÓN

Roll: -128.30 °
Pitch: -6.56 °
Yaw: 73.19 °
Latitud:
Longitud:
Neutro

PROPULSORES

Propulsores verticales:
Propulsores horizontales:
Intensidad de los LEDs:

CALIDAD DEL AGUA

PH: 7.7 und
Conductividad: 55400 uS/cm
Turbidez: 7.6 NTU
Oxígeno disuelto: 50.1 %Sat
Temperatura: 21.42 °C
Profundidad: 0.64 m
Salinidad: 36.8 PSS

CALIBRACIÓN

Figure 2.6: ROV interface

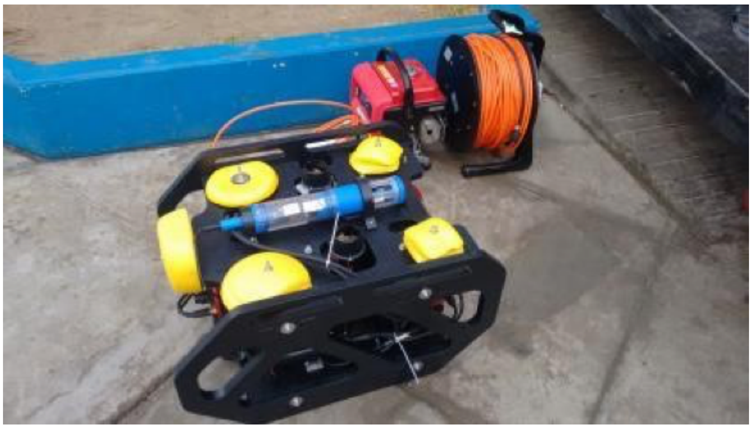


Figure 2.7: ROV assembled

2.3.2 Multiparameter Probe

The water quality sensors were selected according to the requirements of the IMARPE to include common parameters for a general overview of water conditions. In this regard, the parameters selected for the probe were: temperature, dissolved oxygen, specific conductance, salinity, pH, depth, and turbidity. The specifications of the selected multiprobe are presented in Table 2.2.

2.3.3 Hydrophones

The hydrophones were selected according to their bandwidth, sensitivity, and frequency response. Underwater noise from engines, one of the most common sources of underwater noise, is mostly propagated in low frequencies. As it is of interest to perform studies on mammal, such as dolphins, which use higher frequencies for communication, the hydrophones required

Table 2.2: Multi-parameter probe sensors

Item	**Specification**
Temperature	-5 to 50°C
Dissolved Oxygen	0 to 50 mg/l
Specific Conductance	0 to 100 mS/cm
Salinity	0 to 70 Practical Salinity Scale (PSS)
pH	0 to 14 units
Depth	0 to 100m
Turbidity	0 to 3000 Nephelometric Turbidity Units (NTU)

bandwidth from 1 to 170 KHz. The selected hydrophone specifications are shown in Table 2.3.

Table 2.3: Hydrophone and amplifier parameters

Item	**Specification**
Usable Frequency Range	1Hz to 170KHz
Receiving Sensitivity	-211 dB ± 3dB re 1V/uPa
Operating Depth	700 m
Custom Hydrophone Amplifier	
Adjustable gain	Up to 60 dB
Low pass filter	None, 0.1KHz or 30 KHz
High pass filter	None, 0.1Hz or 10 Hz
Digital sampling frequency per channel	Up to 1MHz

The output is in low powered voltage sensitive to noise; therefore, it has to be conditioned (amplified and filtered). A custom-made configurable conditioning electronic board with serial communication is used for this purpose. The parameters are set by the embedded computer before logging. Gain can be set up to 60 dB, and there are fixed filters for configuring the board: 0.1 or 10 Hz high-pass filters and 100Hz or 30 KHz low-pass filters. The amplifier is designed with two amplifications steps before the Programmable Gain Amplifier (PGA) with a total gain of 13db to assure a signal in the order of millivolts. Fig. 2.8 shows the block diagram of the custom hydrophone amplifier where the selectors for filters and gain are observed within the digitalization process.

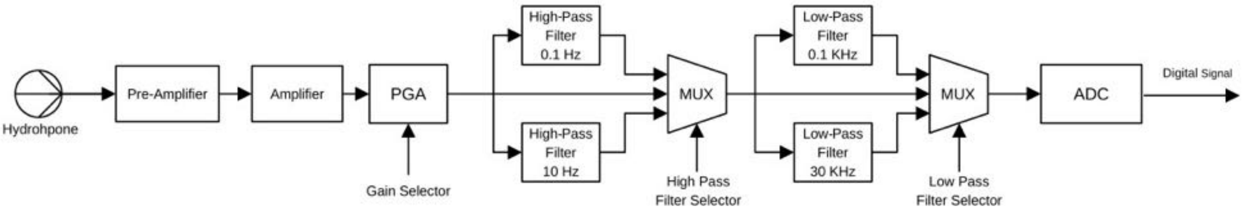


Figure 2.8: Block diagram of hydrophone amplifier

2.4 Preliminary Tests

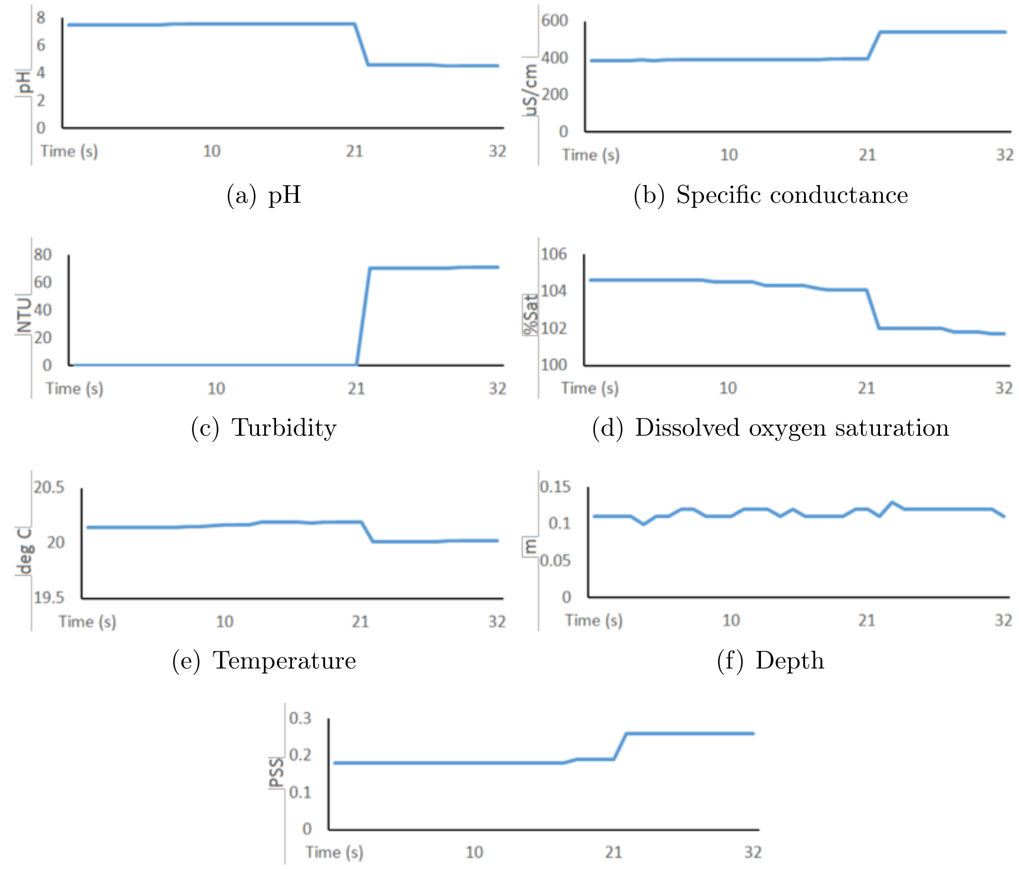
2.4.1 Indoor Tests

The ROV sensors acquisition was tested in a 3m diameter pool and 1.5m depth. The interface and logging were tested to record data of the multiparameter probe and the three hydrophones.

The setup for the multi-parameter probe is due to test the communication between the probe, the ROV and the online visualization of data. The probe is placed in a six liter container with freshwater where substances such as chlorine and bottled orange juice are added to alter the parameters. The results of the variations are shown in Fig. 2.9. The normal conditions of freshwater and chlorine are logged for the first 20 seconds. Clearwater has a turbidity of 0, temperature of 20ºC and neutral pH. Then 300ml of cold orange juice is added at the time of 21 seconds, where, as expected the transition is clearly notable with a regular drop in acidity (from 7.58 units to 4.55 units), a high rise in turbidity (from 0 NTU to 70.9 NTU), a small decrease in dissolved oxygen (from 104.6% to 101.7% of saturation) and a small increase of salinity (from 0.17PSS to 0.26PSS).

The hydrophones were mounted in a triangular array on the ROV with the thrusters switched off to avoid additional noise. The test consisted of monitoring the acoustic signal of a small spherical object dropped in the pool at 1m distance from the array. There is one hydrophone (CH1) closer to the object, and the other two are symmetrically distributed behind the front hydrophone.

Fig. 2.10 shows the result of the test sampled at 1MHz by the ROV, as expected CH1 is first stimulated followed by CH2 and CH3 which react almost in synchronization, phase around $2ms$ and $2.5ms$. The peaks of pressures are around $120Pa$, the chart shows the compression and expansion of the medium caused by the object.



(a) pH

(b) Specific conductance

(c) Turbidity

(d) Dissolved oxygen saturation

(e) Temperature

(f) Depth

(g) Salinity

Figure 2.9: Multiparameter probe test

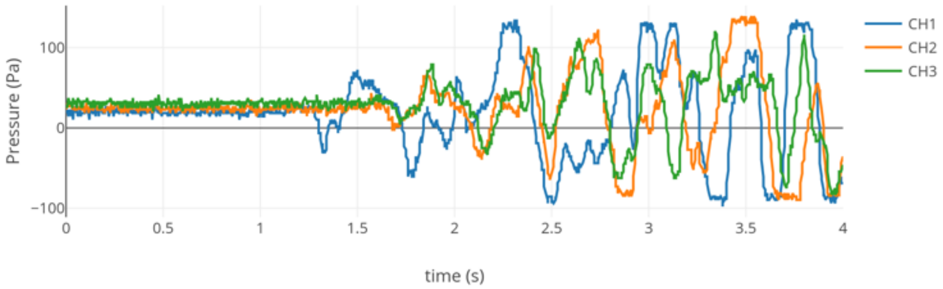
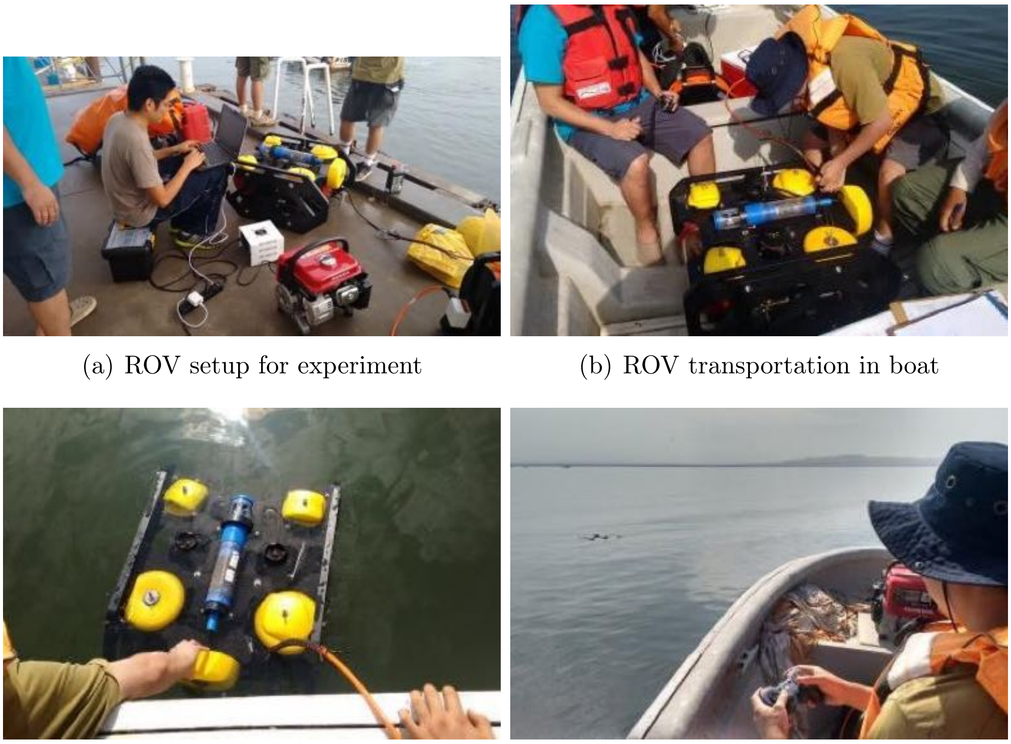


Figure 2.10: Hydrophone array test

2.4.2 Field Test

To evaluate the navigation capabilities of the ROV, the robot was tested in the open sea. The environment chosen for the assessment was the National Reserve of Paracas, located in the department of Ica at the South of Peru. This national reserve, rich with marine resources, provided clear water free of marine traffic to perform the navigation tests, acquire water quality parameters and capture images and video from the underwater camera.

Fig. 2.11 shows the procedure execution of the field test. The ROV's buoyancy was adjusted to achieve a slightly positive buoyancy in water, which could be reached simply thanks to the removable buoys.



(a) ROV setup for experiment

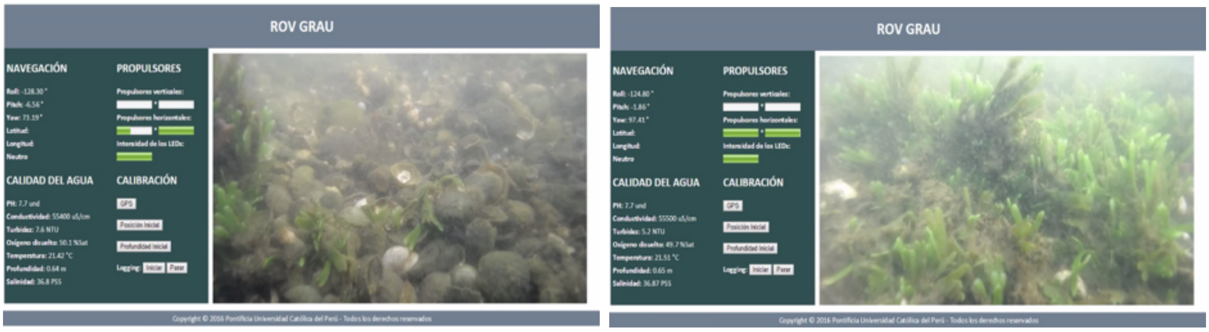
(b) ROV transportation in boat

(c) ROV deployment

(d) ROV operation

Figure 2.11: Experiment execution

Researchers of SERNANP guided the team to a shallow area next to the town of Paracas. The ROV was operated from a boat at around 2.6Km from the shore in open sea. The area was populated by a significant number of scallops, and it was of interest for this institution to perform monitoring of these marine species. Illumination and turbidity in this zone were favorable to perform video recording. Fig. 2.12 shows two screenshots of the ROV interface when exploring the scallops area and surrounding areas with the presence of algae.



Roll: -128.30 °
Pitch: -6.56 °
Yaw: 73.19 °
Latitud:
Longitud:
Neutro
Propulsores verticales:
Propulsores horizontales:
Intensidad de los LEDs:
PH: 7.7 und
Conductividad: 55400 uS/cm
Turbidez: 7.6 NTU
Oxígeno disuelto: 50.1 %Sat
Temperatura: 21.42 °C
Profundidad: 0.64 m
Salinidad: 36.8 PSS
Logging:

(a) Sea Scallops

Roll: -124.80 °
Pitch: -1.86 °
Yaw: 97.41 °
Latitud:
Longitud:
Neutro
Propulsores verticales:
Propulsores horizontales:
Intensidad de los LEDs:
PH: 7.7 und
Conductividad: 55500 uS/cm
Turbidez: 5.2 NTU
Oxígeno disuelto: 49.7 %Sat
Temperatura: 21.51 °C
Profundidad: 0.65 m
Salinidad: 36.87 PSS
Logging:

(b) Presence of Algae

Figure 2.12: Operation screenshots

2.5 Chapter Summary

In this chapter, we have reviewed some of the instrumentation and the surveys performed by the IMARPE in fishery and conservation studies which has been the base of the implementation of a ROV-based acquisition with capabilities of video capturing, water quality parameters measurements and underwater noise measurements.

The ROV platform, the multiparameter probe, and the hydrophones were successfully tested at this stage. The results of preliminary tests of the system show promising applications of the ROV in conservation and environmental studies, as shown during the evaluation of the platform in a real application of species monitoring by the SERNANP research group.

The future work with this robot consists in the proper calibration and evaluation of the sensors by specialists in underwater chemical parameters and acoustic. Additionally, we will propose this platform to institutions such as the IMARPE and SERNANP for fieldworks as a complement to their instrumentation.

Chapter 3

Hardware and Software Framework for Mobile Robots Applicable to Underwater Vehicles

Mobile robots development usually includes the use of controllers, hardware and software interfaces for sensors and actuators, and a simulation environment. Nowadays, many off-the-shelf products can be integrated to deliver a combined framework to meet the developers' need. Sometimes there are gaps in the integration that requires hardware adaptations or software plugins to fully incorporate the frameworks which is time-consuming. We present a comprehensive framework for mobile robots development based on the Eyebot7 IO Board [50], a low-cost expansion board that interfaces with sensors and actuators. It includes an Application Programming Interface (API), Robot Operating System (ROS) integration and a simulation environment for wheeled and underwater robots. Evaluation of the controller through simulations and real robot tests are shown. Finally, the framework is applied to the upgrade of two underwater robots.

3.1 Introduction

Mobile robots are based on the integration of sensors, actuators and a controller through hardware and software interfaces that allows the development of algorithms to control the robot. Ultimately, it is desirable a simulation environment for testing control algorithms before the actual deployment of the robot in a real scenario. Control system structures defined in different layers are commonly used in robotics [51, 52]. These structures allow us to define *low-level* layers in charge of handling sensors, actuators, signal conditioning and filtering, in addition to other tasks that communicate to higher levels responsible for control and monitoring [53]. These *low-level* layers can be software or separate hardware that communicates to a higher layer.

There are different commercial-off-the-shelf expansion boards compatible with standard controllers that allow interfacing sensors and actuators. These interfaces range from simple on-off transistors, relays, analog to digital converters and motor drivers to embedded controllers for vehicle speed control and navigation [54]. Typical expansion boards for robotics offer servomotors control, Pulse Width Modulation (PWM) signal generation, analog/digital inputs and outputs such as Pi-Plates [55], RoboPi [56], IOIO Board [57]. More advanced boards offer autopilot and path planning such as the Pixhawk [58] which is capable of driving robots through waypoints using Global Positioning System (GPS) coordinates and an Inertial Measurement Unit (IMU).

The integration of different technologies sometimes requires hardware adaptations such as little interface boards for signal conditioning and motor drivers, or software plugins and libraries to interface with boards, Robot Operating System (ROS) or to a simulation system. Therefore, we present a fully mobile robot framework based on a low-level interface board, the Eyebot7 IO, which includes serial communication, a C language Application Programming Interface (API), ROS integration, and a simulation environment. The Eyebot7 IO is a low-cost general purpose Input/Output board based on a microcontroller which interfaces with digital and analog sensors and provides PWM control signals to servo motors. It provides four motor drivers for small DC motors and a motion controller for differential drive Wheeled Mobile Robot (WMR). A comparison of common expansion boards including the proposed one is presented in Table 3.1.

In the second part of this section different parts of a mobile robot framework are explained. In Section 3.2, the hardware as well as motion controller are presented. The different layers to interface to the Eyebot 7 IO as well as the simulator are explained in Section. 3.8 and Section 3.4. The tests of the controllers algorithms and the implementation of two underwater robots are given in Section 3.5 and Section 3.6. Finally, the conclusions are presented in Section 3.7.

3.1.1 Related Work

Expansion Boards for Robotics Applications

A common minicomputer used in robotics is the Raspberry Pi (RPi) for which custom boards have been designed such as Pi-Plates [55] and RoboPi[56]. They offer a family of extension boards stackable onto the RPi offering dedicated IO for data acquisition and control. The MOTORplate from Pi-Plates is specific to drive motors, handling DC and stepper motors.

Table 3.1: Expansion boards for robotics applications comparison

Concept	**Eyebot7 IO**	**PIXHAWK PX4**	**IOIO**	**Pi-Plates MOTORplate**	**RoboPi**
Main communication	USB	USB	USB/ Bluetooth	2 GPIO	I2C
GPIO	16 I/O	5	48 I/O	8 inputs	24
Internal Sensors	-	accel, gyro, magnetometer, barometer	-	-	-
DC Motors	8 PWM, 4 Motors	14 PWM	9 PWM	4 Motors	0
Motor current (A)	1	-	-	1.2	0
Encoders	4 Quad	-	6 Pulse Input	4 Single	8 Pulse Input
Control functions	PID speed control, VW navigation control	Geo positioning mission planner. Point to point navigation	-	Open loop	0
Servos/ ESC	14	14	-	0	24
Stepper motors	0	-	-	2	0
Analog inputs	22	2	16	0	8
V input	5-15VDC	5VDC	5VDC	5-15VDC	5 VDC
Firmware upgradable	Yes	Yes	Yes	Yes	Yes
Control libraries	C++	C++, Python, ROS	C/ Python	Python	-
Others	Battery level monitoring, microhpone, 4 LEDs	5 UART, I2C, SPI, 2 CAN, RC. Piezo audio. Control adaptable to common robots	GPIO, PWM, I2C, SPI, UART, Input capture, Capacitive sensing	1 LED, stackable up to 8 boards, 8 core 32 bit RISC MCU @ 100Mhz	0
Approx. Price (USD $)	40	100 - 250	40	40	55

Another common platform for designing robot applications are Android phones for which USB-OTG and Bluetooth communication are exploited in the IOIO Board [57]. This board is basically a breakout board of a PIC microcontroller offering 48 I/O pins, 9 of which are PWM capable, 6 pulse input (suitable for encoder or RC input signals) and 16 are ADC.

The Pixhawk [58] is one of the most popular controllers for robotics, it uses PID controllers for stabilization and navigation with applications in robot cars, boats, copters and underwater robots. Its main focus is to drive brushless motors and servos to match different robot configurations. In the case of brushless motors, it needs external Electronic Speed Controller (ESC). It has a computer interface to configure the board to a specific robot to drive it manually or autonomously through waypoints.

PID controllers are broadly used for motion control [59], in more sophisticated controllers we can still find PID controllers with variations of fuzzy logic for path planning [60], fuzzy logic for self-tuning parameter [61] and other variations [62].

ROS

ROS is an open source *robot operating system* which offers a structured communication layer on top of the host operating system and allows different processes to publish (output data, stream) or subscribe (input data, read) to topics. Processes are called ‘nodes’ in ROS nomenclature, they can be developed to abstract hardware into ROS, to process data or control [63]. It is modular, flexible and encourages the reuse of code in the community such as in hardware abstractions for common sensors. For example, an IMU which has its own protocol, set of settings and parameters, is abstracted as a node which publishes the heading of a robot, which another node can subscribe to it and use it in a control loop. Both nodes are independent and can be interchanged for other nodes.

ROS has become an emerging standard in robot platforms. It has been used in a variety of applications including mobile robots cars and underwater ROVs and AUVs [64–67]. The original paper presenting ROS [63] has more than 4,500 citations, according to Google Scholar 2018-02-12, which is a indicative of its popularity nowadays.

Mobile Robots Simulators

There is a large amount of robotics simulators for mobile robots such as Gazebo [68], ArgOS [69], Webots [70] and ROS+Rviz [63]. In the case of underwater robots, there are some exclusive simulators such as UWsim [71], Subsim [72], and MarineSIM [73], other general purpose simulators such as Gazebo include plugins to extend its capabilities to simulate underwater robots.

Physics and graphics engines are commonly used to provide the functionality and visualization to the simulators. As an example Gazebo incorporates ODE [74], which provides models for rigid bodies dynamics including collision detection, mass and rotational functions desired in mobile robots [75]; and OpenGL Utility Toolkit (GLUT) (OpenGL Utility Toolkit), as well as OGRE [76], for visualization, which renders the robots, the scenario, and the objects. Another exploited alternative for mobile robot simulation is the use of game engines. This also includes physics, rendering, and other engines to develop virtual scenarios [77].

Virtual sensors and actuators are added in libraries which contain the models to interact with the objects and the scenario. The actuators give motion to the bodies, and the physics engine handles its interaction with the environment. Sensors abstract the simulation scenario through a particular model to provide simulated readings.

3.2 Eyebot7 IO Board Structure and Motion Controller

The board design is based on common needs for robotics applications and the boards available in the market such as the boards mentioned in Section 3.1. One of the main objectives of the board is to create a simple command set to communicate with a computer. Another important aspect is the need to have inbuilt algorithms to drive a two-wheeled differential drive car since these robots are broadly used in academia, research, and industry.

3.2.1 Hardware Overview

The Eyebot IO is designed around the Atmel ATxmega128A1U microcontroller. It also provides polyfuse protection, multi-voltage input supply, 5 volts software controlled output (ideal to powering a mini computer), a microphone, 4 full H-bridge motor controllers, and jumpers to select the voltages for the DC motors and servos. A block diagram is shown in Fig. 3.1.

3.2.2 Motion Controller

The board has three inbuilt functions for controlling a set of motors based on the feedback of encoder sensors on each motor. The functions are independent motor velocity controller; velocity-omega (v-w) controller, for controlling a differential drive WMR and; a position controller, that allows a WMR to drive straight, turn and curve for a defined distance or angle.

PID Velocity Controller

Each motor has an independent velocity controller based on the readings from the quadrature encoders. The PWM control signal for the motor is calculated in Eq.3.1. Each of the PID parameters is configurable via command set. The error corresponds to the difference between the current velocity and the desired velocity, $error_{old}$ is the previous error and, $error_{old2}$, the predecessor.

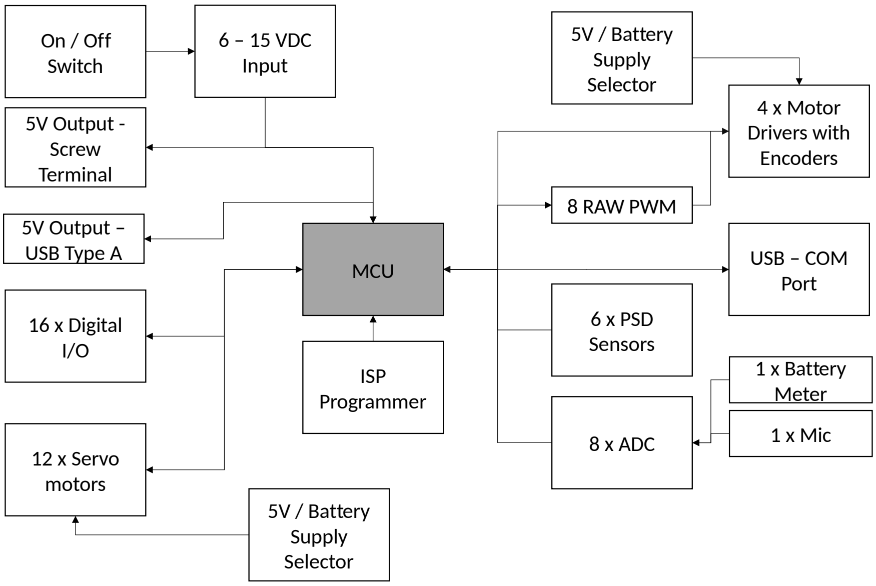
$$\begin{aligned}\text{PWM} = \;& PWM_{old} + Kp * ((error - error_{old}) + \\ & Ki * (error + error_{old})/2 + \\ & Kd * (error - 2 * error_{old} + error_{old2})\end{aligned} \tag{3.1}$$

v-w Controller: Design and Simulation

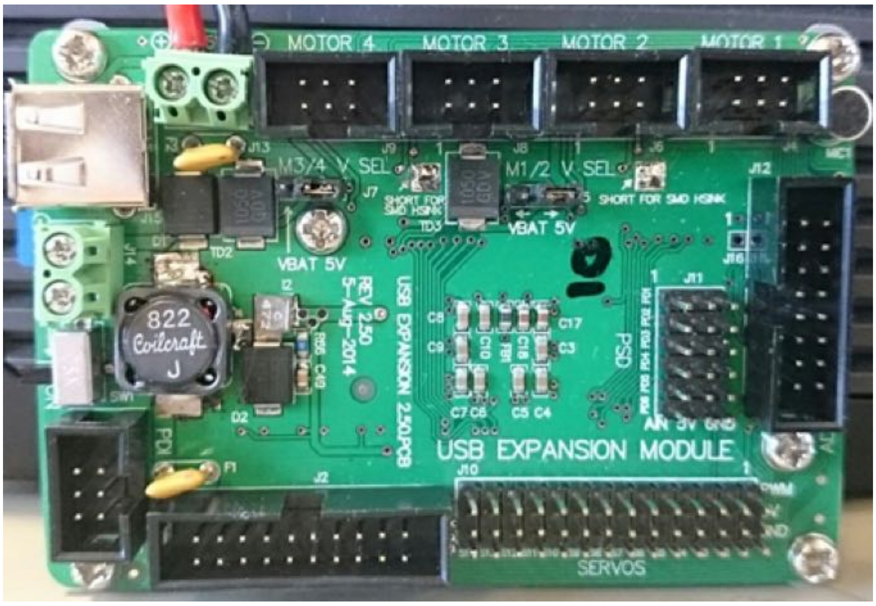
A natural approach to control a wheeled differential drive car is to tackle the problem from the velocity (v) and angular velocity (w) of the vehicle instead of analyzing each wheel speed independently.

In Fig. 3.2 (a) a robot model is shown where r is the wheel radius and L, the distances between wheels. The kinematics and dynamics for control purposes of these cars have been broadly explored such as in [53]. In [78] the v-w model is deducted having the independent wheel speeds v_l and v_r based on the desired v and w of the vehicle. In Fig. 3.2 (b) an Autonomous Underwater Vehicle (AUV) with similar configuration is shown.

$$v_l = \frac{2 * v - w * L}{2 * r} \tag{3.2}$$

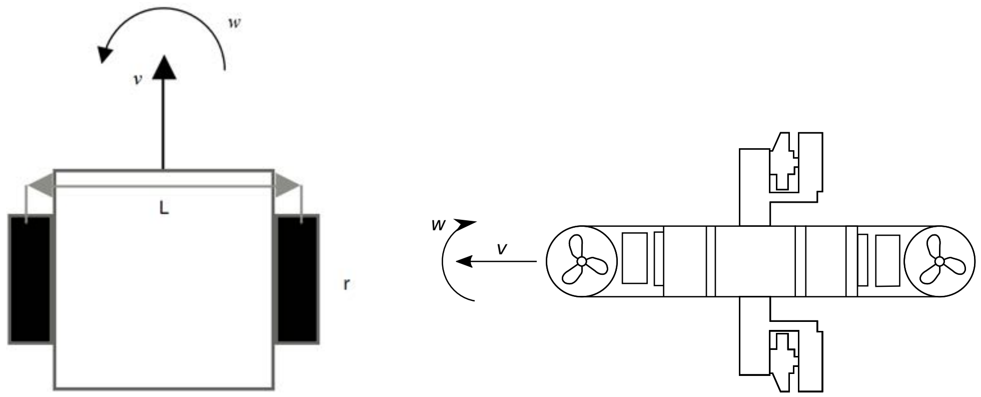


(a) Block Diagram



(b) Physicall Board

Figure 3.1: Eyebot7 IO Overview



(a) Two wheeled differential drive robot

(b) Four thruster AUV

Figure 3.2: Differential drive/thrust robots

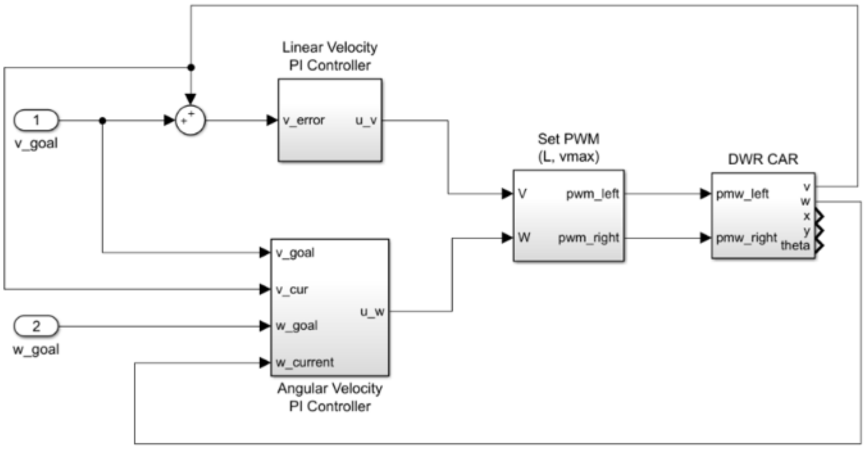


Figure 3.3: $v - w$ Controller Diagram

$$v_r = \frac{2 * v + w * L}{2 * r} \tag{3.3}$$

The PID speed controller can be used independently for v_l and v_r for driving the car and maintain the desired speed for each wheel. Additionally, a v-w controller is defined to control a differential drive robot using v and w as inputs and the PWM signals for each wheel as outputs as shown in Fig. 3.3. Where there is a desired v as an input to the controller the w_{error} is calculated as $(w_{goal} * v/v_{goal}) - w$. Otherwise, it is calculated as $w_{goal} - w$ as in the case of the linear velocity.

For simulating the controller, a simplified parametric model of a WMR for v and w is used [79]. The parameters are obtained using the MATLAB system identification toolbox for different combinations of PWMs for the drive straight, curve and turn commands. The comparison between the real robot behavior and the identified model is shown in Fig. 3.4.

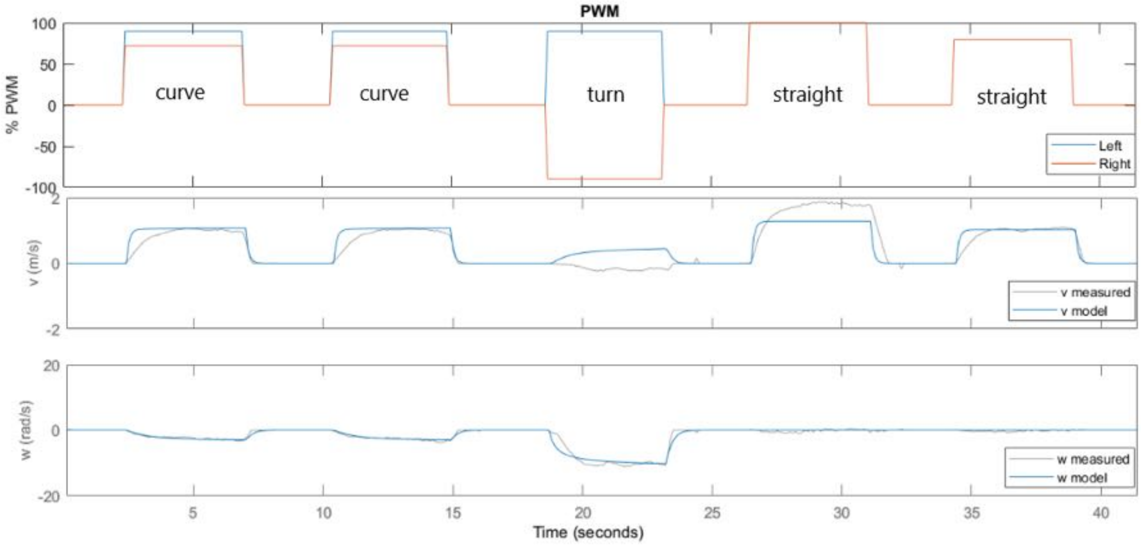


Figure 3.4: Identified model simulation and measured data from a real WMR car

Then, the $v - w$ controller is tuned and simulated for desired velocities and angular velocities. The results are shown in Fig. 3.5 for driving straight, turning and curving. The control signals (PWMs) are also displayed.

Driving Functions

In order to be able to drive the robot from a high-level interface three driving functions are defined: *Drive straight*, with cruise velocity and desired distance as inputs; *Drive turn*, with cruise angular velocity and desired rotation angle as inputs and; *Drive curve*, with a curvature radius R, the length of the arc l and linear cruise velocity as inputs.

The Eyebot7 IO implements a velocity profile controller as shown in Fig. 3.6 in order to generate the desired speeds for the *v-w Controller* to reach the desired distance, angle or arc [78]. Simple navigation planning is implemented through a velocity profile [80]. There are four stages: A, where a constant acceleration is defined until it reaches the desired cruise velocity; V, where the robot is driven at a constant velocity; D, where the robot starts decelerating linearly and; S, where the robot performs a damped deceleration to not overshoot.

A simplified explanation of the code for linear velocity is shown in Fig. 3.7. v_curr_goal is the output of the navigation planner and it sets the references to the $v - w$ controller. The inputs are v_drive_goal (linear distance to drive) and v_goal (cruise velocity). The feedback is provided by v_is (current velocity) and v_driven (distance traveled). $decel\_v$ and $accel\_v$

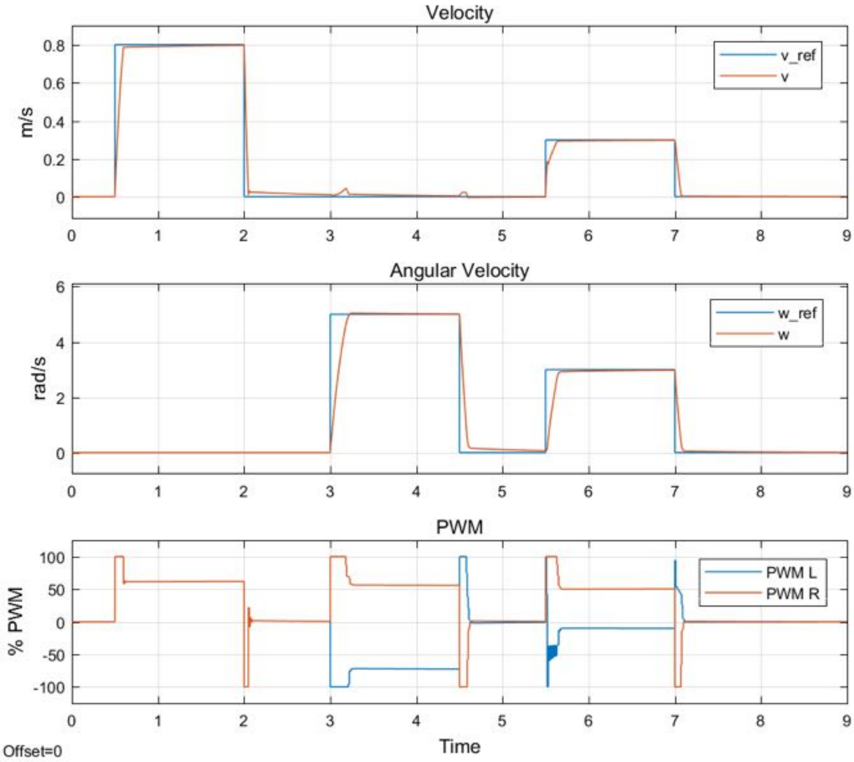


Figure 3.5: Simulation of the $v - w$ controller

are the deceleration and acceleration constants. It worth mentioning that the controller is executed at 100Hz. Therefore, the constants are scaled to it. The code predicts the deceleration of the robot by integrating the current velocity and comparing this with the current state to set the goal. When the robot is decelerating and the predicted deceleration does not meet the expected result, a forced decelerating is applied.

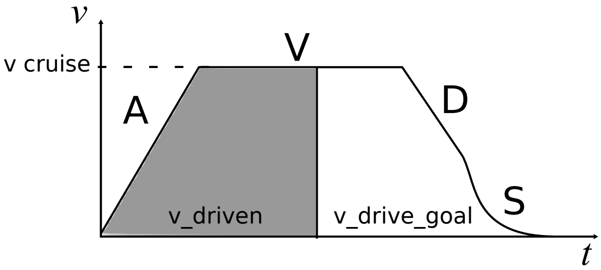


Figure 3.6: Velocity profile applied to v and w

Similar to the *v-w Controller*, the w_{goal} is a function of the v_{goal} and $v_{current}$. For the first two driving functions, the parameters are parsed directly to the controller. In the case of *Drive curve* function, the relationship ($v = w * R$) is exploited in order to generate the set of v and w

```
if ($v\_is * v\_is/(200 * decel\_v) + v\_driven - v\_drive\_goal >= 0$) then
    $decel\_v\_bool = 1$;    {Decelerating ramp}
    $v\_curr\_goal+ = -decel\_v$;
else if $decel\_v\_bool$ then
    $v\_curr\_goal+ = -1/100 * v\_is * v\_is/(2 * (v\_drive\_goal - v\_driven))$;    {Force deceleration}
    if $v\_curr\_goal - 0.0001 < 0$ then
        $v\_curr\_goal = 0.0001$;    {Minimum speed}
    end if
else if $v\_goal - v\_goal > 0$ then
    $v\_curr\_goal+ = +accel\_v$;    {Accelerating ramp}
else
    $v\_curr\_goal = v\_goal$;    {Constant velocity achieved}
end if
```

Figure 3.7: Simplified code for the velocity section of the $v - w$ controller

to accomplished the goal.

Other utilities of the *v-w Controller* include retrieving or setting the pose of the robot via command, changing the PI constants of the controllers, querying the board if the driving command has finished and how much distance is left to travel.

3.3 Interface: Command Set, API and ROS

The Eyebot7 IO can be commanded directly through an ASCII command set over a computer, using a C library for control programs and a ROS node as shown in Fig. 3.8

3.3.1 Command Set

The board uses serial communication over USB in two modes: ASCII and binary. In the first mode, commands and responses are in ASCII. Optionally, it also returns extra information for making it easy to read and to debug possible errors. In the binary mode, the commands and responses are in binary to reduce the communication bandwidth.

An example of the command set used in ASCII and binary modes is presented in Table 3.2. In the table, the command 'm' is used to set the desired % of PWM applied to a specific motor. In a similar way, 's' is for setting the desired angle (scaled from 0 to 255), and 'e' returns the value of a counter associated to an encoder. The full list of commands can be found at `http://robotics.ee.uwa.edu.au/eyebot7/IO7.html`.

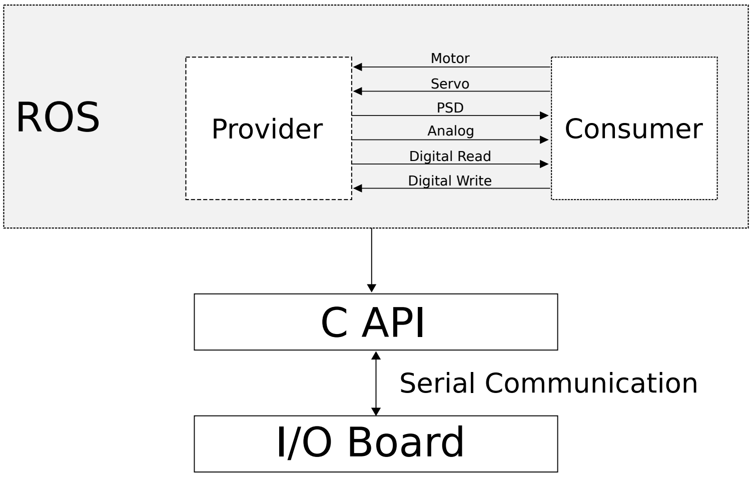


Figure 3.8: Eyebot7 IO board interface

3.3.2 API

The API abstracts all the binary commands to be used as a high-level interface from a controller such as a computer (Fig. 3.8). It is written in C language and is part of a set of libraries to operate on a RPi with a Liquid Cristal Display (LCD) touchscreen for user operation. Examples of the API commands are presented in Table 3.2. A complete description of the API can be found at `http://robotics.ee.uwa.edu.au/eyebot7/doxygen/html/`

3.3.3 ROS Integration

The ROS integration is achieved by a package to interface the board with ROS Core. It offers two classes, the *Provider* and *Consumer* which can communicate through standard ROS messages (Fig. 3.8) an lets you to subscribe and unsubscribe from topics on run-time. The library and package can be found at `http://robotics.ee.uwa.edu.au/eyebot7/`

In the development of a control algorithm on top of the ROS package, a *Provider node* and a *Controller node* have to be created based on the classes presented above. Since the board is able to receive commands and to stream data, the *Provider node* is subscribed to topics such as *Motor* and *Digital Write* to receive commands, and it publishes data to topics such as *Digital read* or *Analog read.* In a complementary way, the *Controller node*, publishes the commands and is subscribed to read the streamed data.

Table 3.2: Example of ASCII binary, API and ROS commands

ASCII Command	Binary Command	API Command	ROS Command
Motor			
m 1 50	<m*><0x1><0x32>	`MOTORDriveRaw(1, 50)`	Controller→setMotor(1, 50)
m 1 100	<m*><0x1><0x64>	`MOTORDriveRaw(1, 100)`	Controller→setMotor(1, 100)
m 1 0	<m*><0x1><0x0>	`MOTORDriveRaw(1, 0)`	Controller→setMotor(1, 0)
Encoder			
e 1	<e*><0x1>	ENCODERRead(1)	Controller→getEncoder(1)
Servo			
s 1 128	<s*><0x1><0x80>	SERVOSetRaw (1, 128)	Controller→setServo(1,128)

Note: In *binary Command* each byte is represented by < >, and the * means the ASCII code for the letter plus 128, which indicates to the board that it is a binary command and notan *ASCII Command*

3.4 Simulator

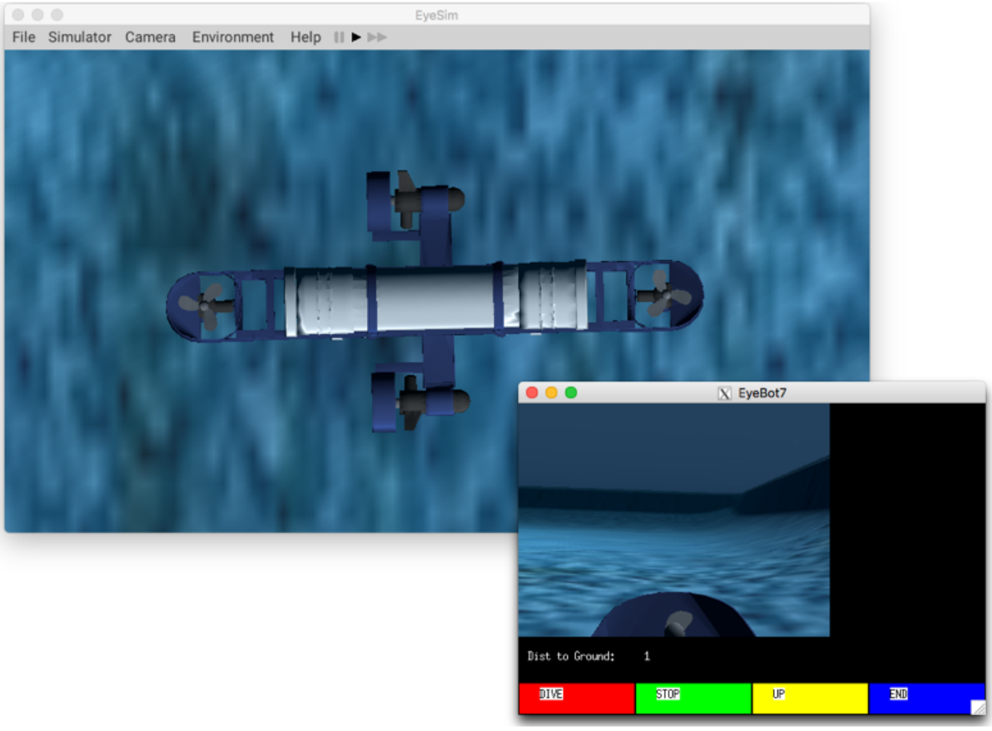
The proposed simulator for the Eyebot7 framework is the EyeSim VR [1] which is based on the game engine Unity [81]. It combines its predecessors EyeSim and Subsim in this new platform.

It is a multiple mobile robot simulator with VR functionality that allows experiments with the same unchanged EyeBot programs that run on the real robots. EyeSim VR is capable of simulating all significant features of the Eyebot IO, including servos and DC motors handling, the v-w controller and analog distance sensors. Additionally, it offers a virtual video camera and an LCD Output/Key Input.

Different scenarios such as a maze, a robot soccer field, and a pool have been already integrated into the simulator providing flexibility to incorporate more environments. In the same way, wheeled robots with sensors and a camera have also been modeled. Underwater robots such as the MAKO and USAL (described in Section 3.6) are also part of the robot family. In Fig. 3.9 the MAKO robot is simulated in a pool, it has a monocular camera displayed on an LCD screen.

In Fig. 3.10 and Fig. 3.11 the results of a simulation of the MAKO are shown. The thrusters are mapped to the "motors" output of the Eyebot7, and the robot is programmed to perform different actions for five seconds and then stop for other five seconds. Following this pattern, the robot is commanded to dive forward, backward, turn right, turn left, and to sink actuating over the corresponding thrusters. In the beginning, the robot is placed closed to the bottom of

[1] `http://robotics.ee.uwa.edu.au/eyesim/`



Dist to Ground: 1

Figure 3.9: MAKO simulation on EyeSim VR

the pool, since it has a slightly positive buoyancy the robot buoys until its equilibrium reached at time 10s (Zr). In "Forward" (time 15s - 20s) the orange plot (Yr) increases as expected, and then the inertia of the robot keeps pushing it forward even after the thrust is applied, the response is similar for "Backward" (time 25s - 30s). For "Turn right" (time 35s - 40s) and "Turn left" (time 45s - 50s) the change is observed in the angle "Phi". For "Sink" the depth (Zr) decreases while the thrust is applied and returns to its buoyant equilibrium after.

The simulator offers a realistic simulation of the motion of the robots and the interaction between robots and objects/walls. Motion and sensor errors can be altered to introduce an error function or noise can be added to make the simulation more realistic.

3.5 Experiments

The Eyebot7 IO is tested in a differential drive robot car to test the integration of the board with external hardware such as sensors, actuators and its usage through a higher level controller. These capabilities are common in any robotics application such as ground robots, drones and underwater robots. Additionally, the robot allows testing the driving functions since each wheel has a motor encoder attached.

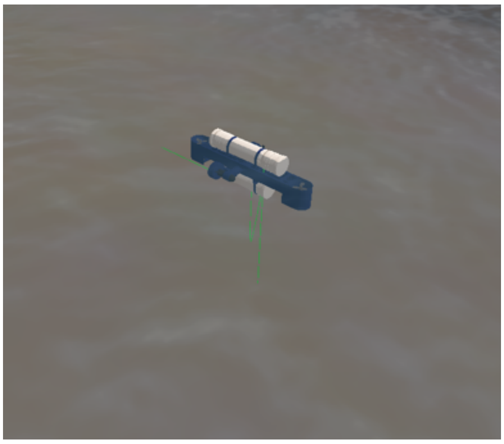


Figure 3.10: Trace of path followed by the MAKO (green line)

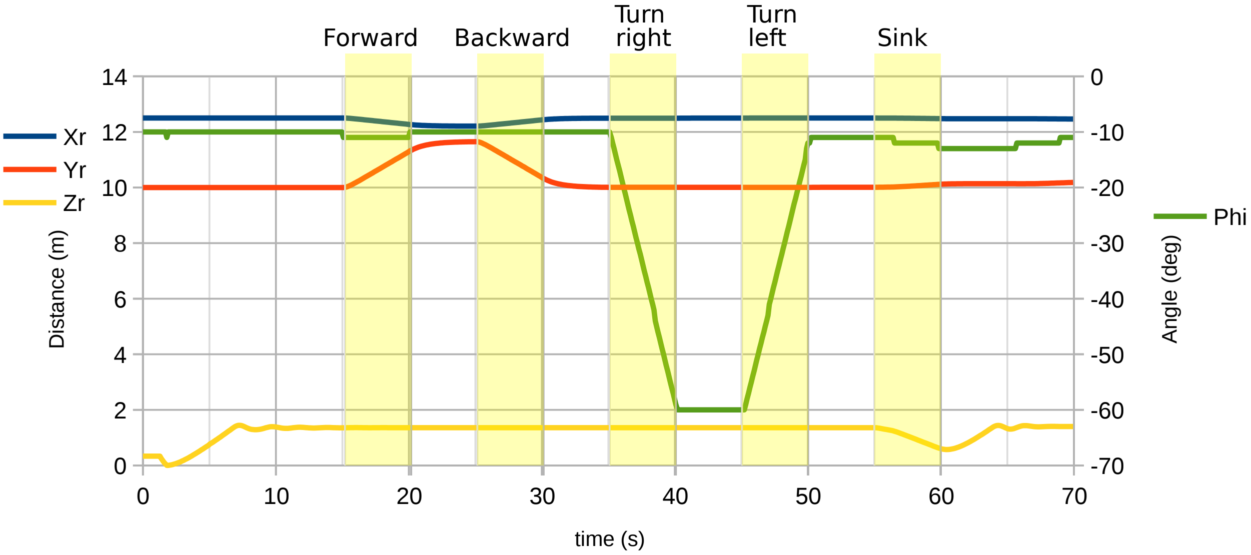


Figure 3.11: MAKO simulation plot

The robot car is equiped with the Eyebot 7 IO and a RPi 3 (stackable on the Eyeboy 7) with a touchscreen display. The robot also has three Position Sensitive Device (PSD) sensors to detect walls and obstacles, a servomotor to actuate a kicker and an electromagnet used to catch small metallic objects. All the external hardware is connected to the Eyebot7 IO, and it is managed by the RPi 3. The experiments were carried out in a wooden coated surface, and the wheels are made of rubber to minimize slippage.

3.5.1 PID Controlled Wheel

One of the PID controlled wheels is tested in the car. The motor drives freely with no load at different desired speeds and a load is then added at timed intervals. The results are shown in Fig. 3.12 where references for 1000, 3000 and 5000 ticks per second are set. Applied load effect can be seen as control signal bumps to control the speed.

The controller regulates the motor PWM as the control signal to the motor. Winding is also considered in the controller limiting the output between -100 and 100 percentage of the duty cycle for the PWM.

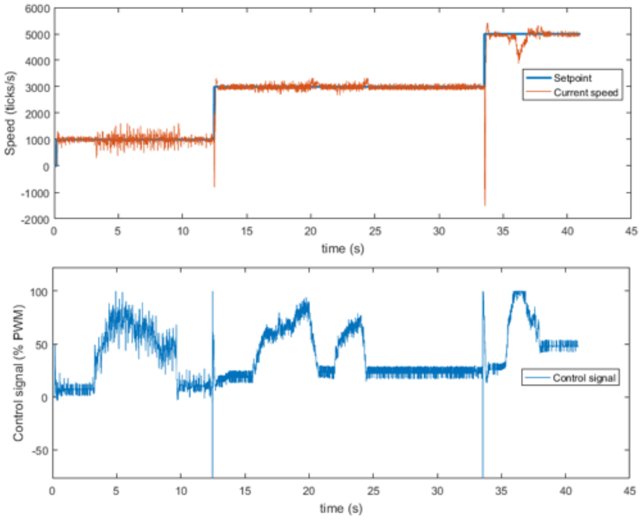


Figure 3.12: Speed PID Controller

3.5.2 Driving Test

A combined experiment is designed to test the driving functions of the v-w controller. The *Drive straight* and *Drive turn* commands are used sequentially to draw a square path. Each of them consists of driving 30 cm straight at a cruise speed of 10 cm/s and turning 90° left at an angular cruise speed of 60°/s approximately (1 rad/s).

Fig. 3.13 shows the results of 12 tests which were obtained by logging the data of the robot's pose acquired through the encoders. It is important to note that the controller has stops bands at +- 2 cm/s for the distance and 0.6° for the angle to avoid overshooting (and the need of reversing to reach the desired goal) and to obtain a fast sequence of commands. It can be seen that there is a shifting in the angle at around 1.5° in every full loop.

A detailed representation of the errors is shown in Fig. 3.14 as a Gaussian probabilistic distribution function (pdf) obtained from the histogram. For the case of the distance error, the

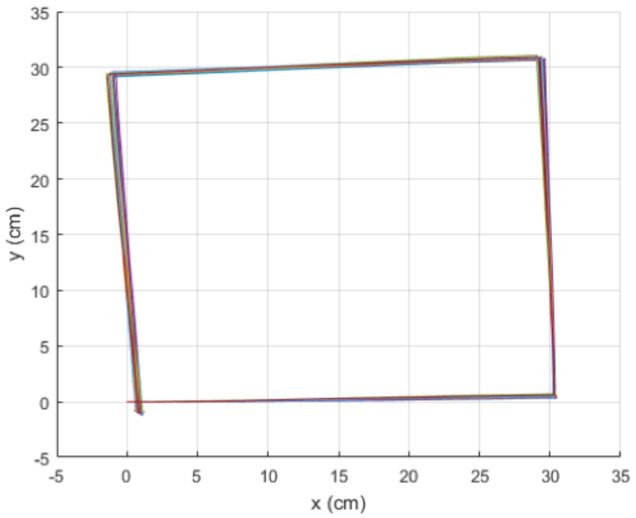
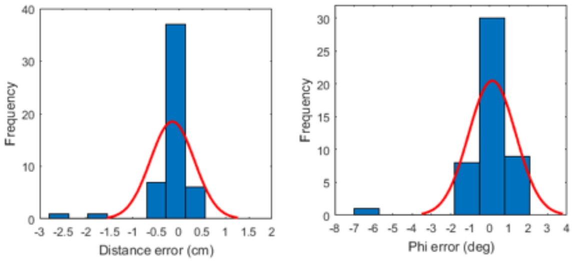


Figure 3.13: v-w square path test. 12 tests running subsequent commands to draw a square

pdf has a mu of -0.1473 cm and a sigma equals to 1.2157 cm. In the case of the angle error, the Gaussian is defined by mu = 0.1352° and sigma = 0.4706°.



(a) Rotation Angle Error (b) Distance Traveled Error

Figure 3.14: Error histograms for $v - w$ controller test

A full loop analysis for the square path is shown in Fig. 3.15 where the PWM output for the motors, as well as the $v - w$ output (read from the encoders) and the angle *phi* are shown. It can be seen four series of the commands *Drive straight* and *Drive turn* to complete the square. When the car was driving forward, both motors tended to run at the same PWM, the small offset is to balance the minimal constructive motor differences and electronics from the feedback of v and w. In case of the turning, both motors rotated in opposite directions, but not with the same magnitude. The *phi* angle tended to be constant for the driving forward sections and had an overdamped behavior to reach the required angle.

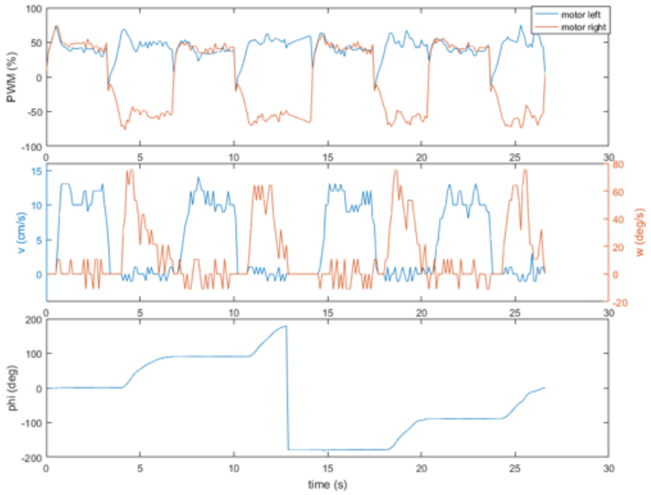


Figure 3.15: $v - w$ single loop analysis

3.6 Underwater Robots Implementation

The Robotics & Automation laboratory at the University of Western Australia has three underwater robots for research purposes. The BlueROV2, from Blue Robotics, (Table 3.3) and two AUVs designed at the laboratory, the USAL and the MAKO. The BlueROV2 uses a RPi 3 and a PIXHAWK board for navigation and sensors reading.

A RPi 3 and the Eyebot7 IO board were used to upgrade the two AUVs. Fig. 3.16 shows a connection diagram for the USAL. The board drives a small pump motor which is connected directly (M3), in case of higher power motors, the PWM signals generated are used to drive a more powerful motor drive (M1, M2). The sensors are also connected to the expansion board, as well as two servomotors. A camera and an IMU are connected directly to the main controller.

3.6.1 MAKO

The MAKO is an AUV based on two cylinders aligned vertically that contains most of the electronics sealed in the top compartment, and the batteries in the bottom. This configuration lowers the center of mass from the buoyancy center giving the robot high stability in water. The electronics boards and batteries are placed on racks that slide into the cylinders to ease the accessibility to the components.

The robot was built in 2004 and was upgraded with the motors and the hulls were replaced, as well as the controllers. It has four thrusters symmetrically placed to perform surge and heave linear motion, as well as pitch and yaw rotation.

It has 4 Ultra Sonic (US) facing the front, sides, and bottom for wall, floor or obstacle

Table 3.3: Overview of BlueROV2 from Blue Robotics [82]

Overview		
Weight	11 Kg	
Size	0.46 m x 0.34 m x 0.25 m (L x W x H)	
DOF	5 (surge, heave, sway, yaw, roll)	
Max Depth	100 m	
Power	270W LiPo Battery	
Actuators		
Thrusters	6 (4 vectored, 2 vertical)	
Lights	2	
Sensors		
Cameras	1	
Navigation	IMU, depth sensor	

detection. It has a pressure sensor that exploits the direct proportionality between the hydrostatic pressure measured and depth in underwater environments. A paddle wheel encoder is used as a velocity sensor. A GPS, which can be used while the robot is on the surface, and an IMU, primarily used for orientation, are also included. A camera is located in the front looking forward. Additionally, there is a USB camera in a housing that can be placed at other angles. An overview of the technical details is shown in Table 3.4.

3.6.2 USAL

The USAL is a torpedo-shaped AUV rated for 15 m depth. It has a stern and heave motor thrusters, a bow thruster (implemented by using a mini pump), and rudder, controlled by a servo motor, for the motion. They allow it to move forward and backward, up and down, rotate and surge in an left or right angle.

It has similar sensors as the MAKO, excluding the US echosounders. Instead, it uses the PSDs as optical distance sensors, which work reasonably for distances lower than 50cm. A camera is mounted in a bracket controlled by a servo for tilt rotation. Technical details of the USAL are shown in Fig. 3.16,

3.6.3 ROS Integration

A ROS setup for the underwater robots is implemented through the ROS package described in Section3.3. The setup is divided in two computers; the RPi, which is inside the robot, and an external computer, which sends predefined commands to the robot and receives images when

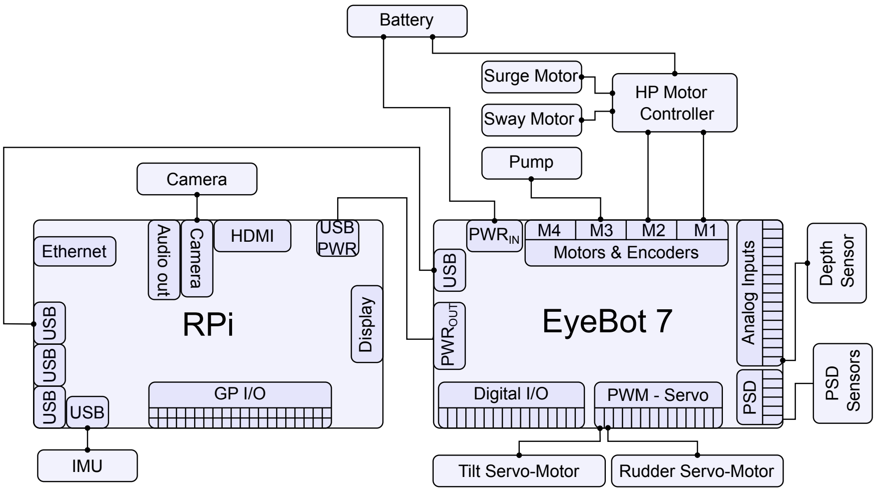
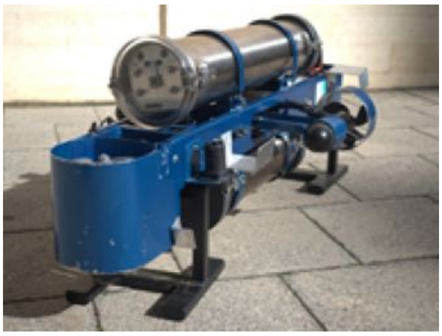


Figure 3.16: USAL electronic connection diagram using an Eyebot7 IO and a RPi

Table 3.4: MAKO AUV: Technical specifications overview



Overview	
Weight	35Kg
Lenght	1.5m
Volume for electronics	∼22000cm3
Stability	High
DOF	4 (surge, heave, pitch, yaw)
Battery	21 Ah
Actuators	
Thrusters	4
Sensors	
Camera	1 fixed position + 1 Optional
Distance	4 US Echosounder
Depth	Preasure sensor
Velocity	Paddle wheel encoder
Heading/ Acceleration	IMU
On surface reference	GPS

Table 3.5: USAL AUV: Technical specifications overview

Overview	
Weight	9Kg
Lenght	0.7m
Volume for electronics	∼8000cm3
Stability	Low
DOF	2 (surge, heave) + surge-yaw
Battery	7 Ah
Actuators	
Thrusters	2
Servos	1
Others	1 Mini pump
Sensors	
Camera	1 Tilt servo controller
Distance	3 IR, placed at the front
Depth	Pressure sensor
Heading/ Acceleration	IMU
On surface reference	GPS

the robot is on the surface and connected to a wireless Local Area Network (LAN) (Fig. 3.17).

Four nodes are launched from the RPi on the underwater robot side: the *Eyebot IO Board Provider*; the *Robot Controller*, based on the consumer class; an *IMU node*, created from the packages available in ROS; and a *Camera node*, which streams the images. On the remote PC the nodes *Display* (for displaying the images from the robot) and *Joy* (for handling a joystick to send instructions) are launched. The nodes publish messages into topics which are read by the subscribers.

3.7 Chapter Summary

The Eyebot7 is a framework designed according to common tasks required in mobile robotics from hardware integration to simulation. It provides an easy communication for debugging and different packages for higher control levels through an API and a ROS package. The ROS package takes it to a new level of interconnection by making it compatible with all other ROS packages.

The simulator environment allows use of the same code as the real robots in the simulator, which makes it easy for testing and developing. The Eyebot7 IO board control loops have been

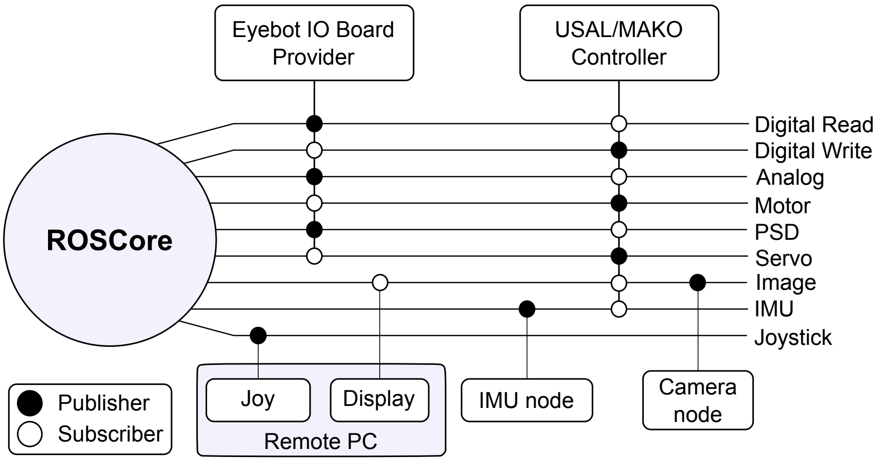


Figure 3.17: Example of ROS implementation for underwater robots

tested showing acceptable results and, additionally, can be tuned from the command line. The driving controllers offer a straightforward interface which allows a user to start driving a car with simple commands. All things considered, the Eyebot7 IO is a low cost and easy to use tool that allows hobbyist or researchers the flexibility to integrate sensors and actuators to a computer as the main controller.

Chapter 4

Underwater Robot Simultaneous Localization and Mapping (SLAM) Frameworks

Localization and mapping are key elements in autonomous vehicles hence robots need to keep track of their position and the environment to trace a path, navigate and avoid obstacles. In the last few decades, different developments in underwater SLAM (Simultaneous Localization and Mapping) have been achieved by three main approaches: Extended Kalman Filter SLAM (EKF-SLAM), FastSLAM, and GraphSLAM. The foundations of these algorithms and their application to underwater scenarios are discussed in this chapter. Furthermore, conventional instrumentation that makes SLAM possible in these situations are also described. Simulation results show how each approach improves localization and mapping for a robot compared to a simple averaging estimation.

4.1 Introduction

Simultaneous Localization and Mapping (SLAM) is a challenging topic for autonomous underwater vehicles (AUV) due to the limitations of subsea localization sensors and exteroceptive sensors for mapping. In underwater scenarios, the use of global positioning systems such as Global Positioning System (GPS) (Global Positioning System) is not possible without tethered floats, since satellite signals are attenuated in underwater environments. Different onboard sensors mounted in underwater vehicles are used instead to estimate an accurate location through data fusion.

The mapping process is performed by sensors capable of perceiving the surrounding environment. Acoustic ranging sensors and imagery from video cameras and acoustic sensors are used. Features extraction, mostly based on image processing, and measurements of the surroundings must be performed for an accurate SLAM.

This chapter is focused on SLAM fundamentals applied to underwater vehicles in marine environments where structured and unstructured scenarios are presented. Different approaches to exploiting scenario features through sensors have been reviewed. Section 4.2 presents instrumentation used in underwater vehicles for localization and mapping. Section 4.3 reviews three fundamentals of SLAM approaches. Section 4.4 implements and compares the approaches reviewed and presents the latest developments in underwater scenarios. Finally, concluding remarks are given in Section 4.5.

4.1.1 Formulation of the SLAM Problem

SLAM is formulated as the iterative process of localization and mapping under the intrinsic dependency between both. A robot creates a map of the surroundings and localizes itself in it. It takes advantage of a known location for building a map and estimating its position, having an updated map.

Two models are defined: the *motion model*, which is used to estimate the location of the robot based on navigation sensors, positioning sensors or motion control signals. And the *observation model*, which is used to abstract the information of the environment through the sensors' perspective and extract significance features such as landmarks. Both models are altered by noise, sensors drift and error accumulation over time [53]. To reduce the uncertainties corrections are made based on the estimation and re-observation of landmarks and robot poses.

SLAM was originated in map building for robots in the early 1990s by Durrant-Whyte and then presented in 1995 at the International Symposium on Robotics Research by Thrun [83]. Other research groups also worked on this topic, such as the Massachusetts Institute of Technology, The University of Zaragoza and the Australian Centre for Field Robotics [84].

Notations

Across the chapter, the following notations are used:

- x_k — robot state vector (position and heading) at a time k
- u_k— control vector to perform motion
- m_i — landmark position matrix

- z_k — observation matrix at time k
- $z_{a,b}$ — specific observation matrix of landmark b at time k

4.1.2 The SLAM Process

In Fig. 4.1 the SLAM process is shown for a 2D representation. There are two landmarks visible for the robot, m_1 and m_i First, the robot observes the two landmarks according to $z_{0,1}$ and $z_{0,i}$, then moves to x'_1 given a u_1. Note that x'_1 is the estimated pose and x_1 the real one. The robot makes a new observation from x_1 and re-observes the landmarks m_1 and m_i which are now being observed through $z_{1,1}$ and $z_{1,i}$. Processing the estimated data from the robot motion and the new observations, a new believe pose for the robot and the landmarks positions, x''_1 and m_1 and m_i (arrows number 4 and 5), are obtained [84].

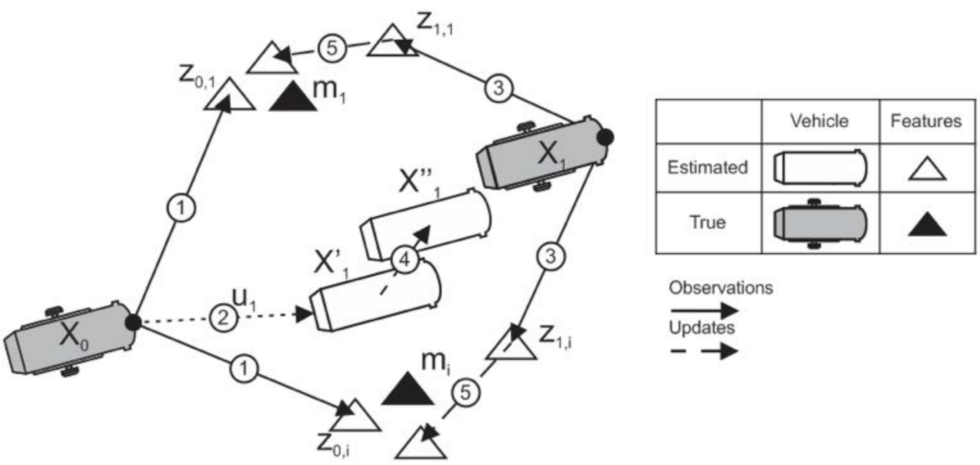


Figure 4.1: SLAM process: (1) Make Initial map z_0. (2) Estimate location given u_k. (3) Make 2nd map z_1 from X_1. (4) Update location given z_1 (5) Update map
based on: Durrant-Whyte et al. [84]

The block diagram presented in Fig. 4.2, by Siegwart and Nourbakhsh in [53] shows the estimation process of data fusion which is the core of SLAM solutions and is covered by state estimation methods [85]. There are onboard localization sensors that include a wide variety of transducers from which data can be combined. The observations in underwater environments are mostly done by acoustic or video imagery sensors. The feature extraction process is responsible for finding landmarks from the exteroceptive sensors. The final matching involves a data association process, where features from the predicted and observed maps are matched to be used later in the update of the map and localization. The features that do not match can be either added as new features in the map, for subsequent iterations, or removed from the map.

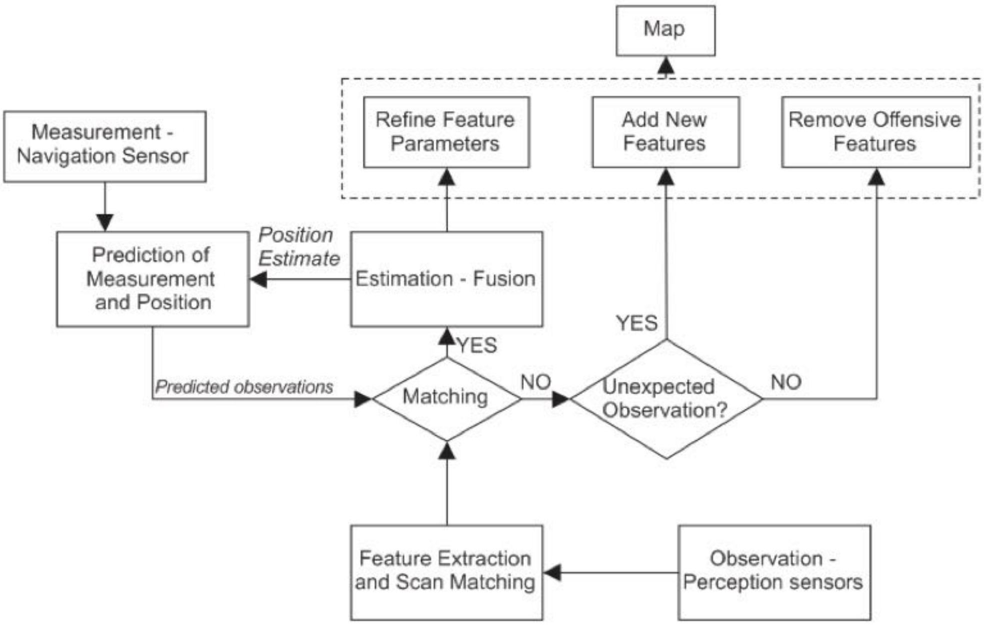

Map
Refine Feature Parameters
Add New Features
Remove Offensive Features
Measurement - Navigation Sensor
Prediction of Measurement and Position
Position Estimate
Estimation - Fusion
YES
YES
Predicted observations
Matching
NO
Unexpected Observation?
NO
Feature Extraction and Scan Matching
Observation - Perception sensors

Figure 4.2: Block Diagram of the SLAM process
Redraw from [53]

There are different approaches for SLAM with the fundamental frameworks being Kalman Filter [84]— particularly the Extended Kalman Filter (EKF)[86]—which can deal with nonlinear models; FastSLAM [87]—which is based on Particle Filters (PF) and also EKF [88]; and GraphSLAM [89]—which abstracts the sensor readings as constraints to build a node graph and updates it iteratively.

4.2 Instrumentation for Underwater SLAM

4.2.1 Onboard Navigation Sensors

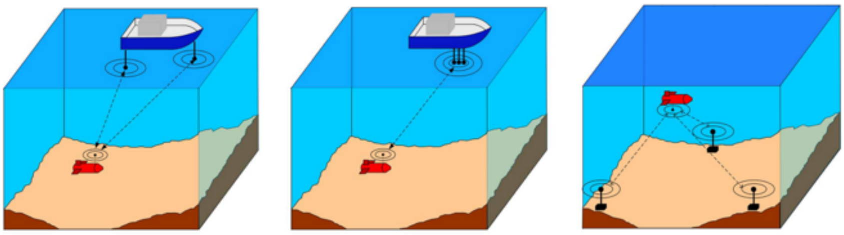
In underwater SLAM the localization update is given by onboard navigation sensors. Depending on the number of sensors, a prior stage for data fusion (based on EKF) can be used [90]. In the integration of sensor readings, it is common to use the first read of the Attitude and Heading Reference System (AHRS)—such as depth sensor and magnetometer—to then integrate Inertial Measurement Units (IMU) readings—such as accelerometer and gyro—or dead-reckoning sensors. AHRS give an accurate heading and reference to integrate the information of acceleration and velocity to obtain an accurate localization estimate [86, 91].

GPS

GPS on its own is unsuitable for underwater surveys, due to the attenuation of the signal when going below the water surface. Even the use of differential GPS has a limited range of around half a meter below the surface [92]. Despite this, it is still used to acquire a fixed position to be able to be used later for dead reckoning (DR) and external sensors to estimate a current position [93, 94]. Additionally, GPS is used to obtain position signals when the vehicle navigates to the surface after a period of time, where it can update its position and reset the accumulated errors from estimation [90]. Depending on the type of GPS technology, prices range from hundreds of US dollars for a standard commercial GPS with an accuracy of around 10 meters, to thousands of US dollars for Differential GPS (DGPS) with post-processing with accuracies of 0.3–2 meters to 0.02–0.25 meters.

Underwater Acoustic Positioning System

Acoustic positioning relies on the basics of measuring the time of flight (TOF) of acoustic returns detected by a set of receivers. Different standard systems vary in the positioning of the transponders (Fig. 4.3) such as Ultra Short Base Line (USBL) or Super Short Base Line (SSBL), Short Base Line (SBL) and Long Base Line (LBL) followed by acoustic modem communication and intelligent systems [95, 96] .



(a) SBL (b) USBL (c) LBL

Figure 4.3: Underwater Acoustic Positioning Systems
from: Paull et.al. [85]

Depth Sensor

Vertical positioning can easily be obtained by pressure sensors such as barometers. The readings of barometric pressure follow the rule: 0.1MPa=10m depth, therefore a direct relation can be used. Surface moving water can add noise to the readings; however, a low-pass filter is enough to eliminate these variations [91]. Due to the steeper gradient of pressure underwater an accuracy of around 0.1m can be achieved at a cost in the range of 100–200 of US dollars [97].

Inertial Measurement Unit—Accelerometer and Gyroscope

IMUs contain a set of three-axis accelerometers and three-axis gyros to provide readings of linear accelerations and angular velocities for the three orthogonal axes. The location is then calculated combining the integration of the linear velocity and the angular velocity. Integrating the velocity with respect to the time for an extended period also accumulates the errors delivering uncertainties of the position after a while [90, 98].

The principle of operation of accelerometers is to measure the force required to accelerate a known mass. Typical configurations are the pendulum, with a bias range of 0.001mg and Micro-Electro-Mechanical Systems (MEMS), with a bias range of 0.01mg. For gyros, the measurements can be carried out by measuring the phase change of laser light through a series of mirrors [Ring Laser Gyro (RLG)] also through different directions using fiber-optic cable, or by using MEMS to measure the Coriolis force in a mass suspended with a spring system. The performance of these sensors is in the range of 0.0001° per hour for RLG to around 60° per hour for MEMS. Prices go up with accuracy from hundreds of US dollars for MEMS to hundreds of thousands of US dollars for optics systems [99].

Magnetometer

The Inertial Measurement Unit (IMU) can also include a three-axis magnetometer that reads the magnetic field of the Earth. Once it is calibrated, this information together with the acceleration and gyroscope are used to estimate the roll, pitch, and yaw angles [94].

Doppler Velocity Log (DVL)

The Doppler Velocity Log (DVL) provides the water velocity readings relative to the Autonomous Underwater Vehicle (AUV) and the velocity vector of the robot fixed to the seabed can then be calculated. The combined data sources from the inertial sensors, depth, and DVL contribute in all to a better estimation [94, 100]. It is used to improve the estimation of localization for underwater robots, the use of DVL can be added to an Inertiail Navigation System (INS).

DVLs typically consists of four beams (in principle three beams are needed for 3-D navigation in a body referenced frame) generated by an orthogonal set of transducers. A Janus configuration is commonly used in which two sensors are angled aiming outside the center, and the other pair is set orthogonally in the same manner [90, 94].

The frequency shift in the reflected signal along each transducer is measured and knowing the configuration angle and distances of the sensors, a system of four equations can be determined and then solved for a set of the three orthogonal velocities. This system is sensitive to misalignments on the transducers and smooth changes on physical properties of water that can vary the propagation speed (depth, temperature, and conductivity/salinity) [101]. The standard deviation of readings from a DVL is in the order of 0.3cm/s to 0.8cm/s, and the cost range is between $20k to 80k USD [102].

4.2.2 Exteroceptive Sensors

In underwater SLAM the sensors gathering information about the environment to build maps are mostly acoustic or video sensors. Video images have good resolution but, even in an underwater controlled environment with good illumination and low turbidity, the range is limited to 60m [103]. Conversely, acoustic sensors have lower resolution, their range depends on the frequencies (the higher the frequency, the higher the resolution, but the lower the range) and they can also work in turbulent water [104]. Fig. 4.4 shows two classifications of acoustic sensors. They are divided into range sonars which provide distances for each sampling, and image sonars which capture images at every scan.

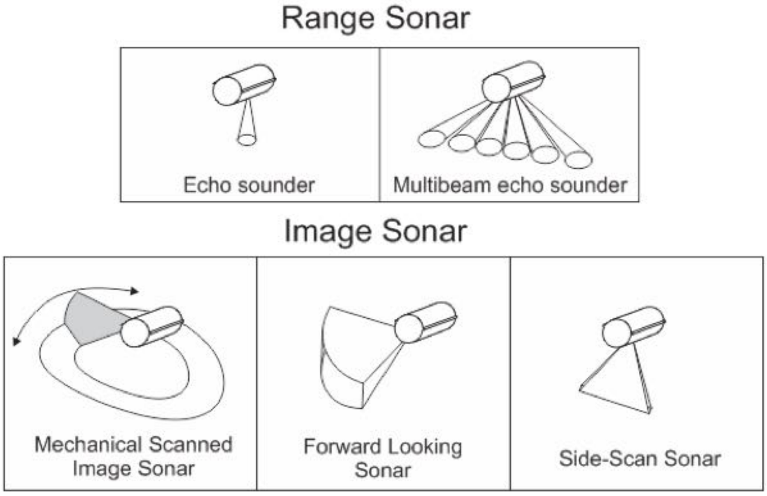


Figure 4.4: Acoustic sensor for perception: Range Sonar, Image Sonar. Redraw from [105]

Multibeam Echo Sounder (MES)

The MES is based on the echo sounder principle where a pulse is emitted by a transducer, it travels through the water and is reflected towards the sensor when reaching a surface. Then the time of flight (TOF) is measured to obtain an estimated distance. The multibeam echo sounder has an array of sensors that emits and receives a fan-shaped beam to the seabed. The result is a bathymetric map of the seafloor [105]. The difficulties of sonar sensors in getting reliable features encourage the use of bathymetric sensors [106]

Mechanically Scanned Imaging Sonar (MSIS)

A mechanically rotated transducer is used to scan a horizontal 2D area. It turns at fixed angles emitting a fan-shaped acoustic beam narrowed at the horizontal and wide to the vertical plane. The intensity of the reflected backscattering is used to build 360° images, and it might take a few seconds to complete a full image [105]. For underwater vehicles, distorted images are collected from the MSIS while the mobile is moving. This has to be taken into consideration in SLAM [107, 108].

Forward Looking Sonar (FLS)

FLS gives an acoustic image of what is in front of the vehicle by emitting a single pulse to an insonified area and then receiving the echo by an array of hydrophones. Since it is possible to get overlap readings Fig. 4.5 (a) when the vehicle is navigating it is possible to set landmarks to re-observe them in the subsequent samples and perform SLAM [109, 110] .

Side Scan Sonar (SSS)

SSS provides downward looking image generation when grouping the readings. Using these images in SLAM is suitable when multi-pass missions are planned (i.e. transects) since the data of each reading does not overlap the previous reading Fig. 4.5 (b). Landmarks can be determined as references for subsequent passes [111, 112].

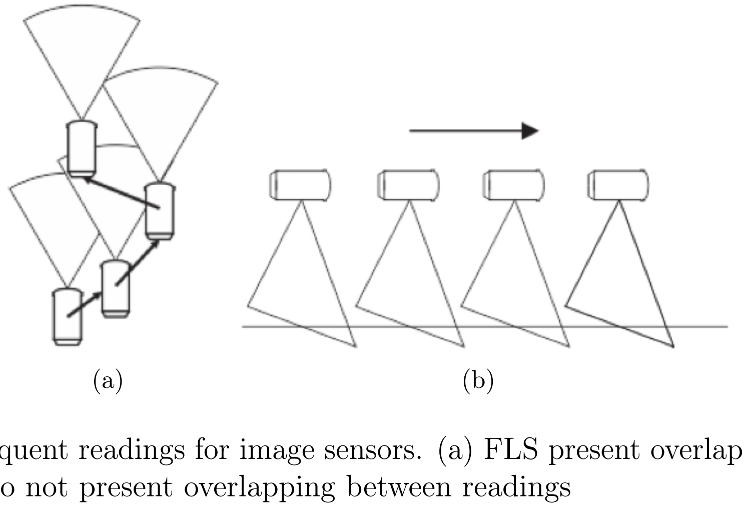

(a) (b)

Figure 4.5: Subsequent readings for image sensors. (a) FLS present overlapping areas between readings; (b) SSS do not present overlapping between readings

Video Cameras

Video cameras provide high-resolution images easy to interpret but limited in ranges of observation due to poor illumination and turbidity in subsea environments [113, 114]. Underwater operations and SLAM using cameras are performed where the vehicle is close to the objective (2-3 m) [115], in applications of hull inspections [116], and in mosaicing of shallow sea bottom [117].

Monocular and Stereo cameras are used where feature detection is carried out to track recognizable landmarks in the ensuing re-observations [23]. Common feature extractors are Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF), methods based on the Euclidean distance of keypoints from a reference image. These methods are used in computer vision [118, 119] and in underwater environments [114]. Spectral methods with multilayer resampling for online 3D mapping are performed in [120] for stereo cameras.

4.3 SLAM Frameworks for Underwater Environments

The three SLAM frameworks equations in this section are based on [84, 121] for Extended Kalman Filter (EKF), [87, 88, 122, 123], for FastSLAM, and [84, 89] for GraphSLAM. The notations of the general approach to SLAM presented are from the work of Durrant-Whyte and Bailey in [84].

4.3.1 Extended Kalman Filter SLAM

Kalman Filter

Mathematically, SLAM can be formulated as a probabilistic function where the robot state vector and the landmarks matrix are obtained given the observations, control inputs and an initial state. We will use the notation and formulation stated in [84].

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0) \tag{4.1}$$

It shows the contribution of the control input u_k and the observation z_k of the landmarks m referred to an initial position x_0, which is arbitrary and often obviated for notation, defined by a pose state vector.

The motion model is represented by:

$$P\left(x_k \left| x_{k-1}, u_k\right.\right) \leftrightarrow x_k = f\left(x_{k-1}, u_k\right) + w_k \tag{4.2}$$

Where $f(*)$ models the vehicle kinematics and w_k, the motion disturbances.

And the observation model:

$$P\left(z_k \left| x_k, m\right.\right) \leftrightarrow z_k = h\left(x_k, m\right) + v_k \tag{4.3}$$

Where $h(*)$ models how the sensor data represents the environment and v_k are the observations.

The above formulas represent the KF approach for linear functions, in the case of Extended KF the concept is expanded to include no linear models. This is achieved by the linearization through Taylor series expansions of the models around the current temporal values [53]. The robot state vector and landmarks are represented as Gaussian distributions having mean and covariance matrixes.

Extended Kalman Filter

The EKF is the nonlinear version of the KF, given the linearization of the models through Taylor series expansion. The filter works under the assumption that the distribution of the values gathered has a Gaussian Model probability distribution minimizing the squared distance for the different sources or samples [53].

The SLAM algorithm is implemented as a recursive motion update (prediction) and observation update (correction). In underwater SLAM the updates are computed from the data obtained by the navigation sensors and the perception sensors.

Motion update / Time update (Predict)

$$\widehat{x}_{k|k-1} = f(\widehat{x}_{k-1|k-1}, u_k) \tag{4.4}$$

$$P_{xx,k|k-1} = \nabla f P_{xx,k-1|k-1} \nabla f^T + Q_k \tag{4.5}$$

$x_{k|k-1}$ is the pose update, where $f(*)$ is the motion kinematics and represents how states change in every step, therefore, the Jacobian Δf is an estimate of how the state will change. P_{xx}, k is the covariance given the pose update and the noise.

Measurement Update / Observation update (Correct)

$$\begin{bmatrix} \widehat{x}_{k|k} \\ \\ \widehat{m}_k \end{bmatrix} = \left[\widehat{x}_{k|k-1}\widehat{m}_{k-1}\right] + W_k[z_k - h\left(\widehat{x}_{k|k-1}, \widehat{m}_{k-1}\right)] \tag{4.6}$$

$$P_{k|k} = P_{k|k-1} - W_k S_k W_k^T \tag{4.7}$$

Where:

$$S_k = \nabla h P_{k|k-1} \nabla h^T + R_k \tag{4.8}$$

$$W_k = P_{k|k-1} \nabla h^T S_k^{-1} \tag{4.9}$$

m_k, represents the map observed given the updated pose, observations and the geometry $h(*)$ and is Jacobian Δh.

The EKF demands a lot of computational effort $O(m^2)$ and it is hard to implement for online applications when having more than hundreds of landmarks. Besides, computing and storing the covariance uncertainties it is not always useful since it may not represent the real error [84, 86, 88, 124].

It also has problems dealing with loop closing (Fig. 4.6), having bad associations due to accumulated errors that cause it to be unable to recognize previous features. There are, however, complementary methods to overcome this problem [125].

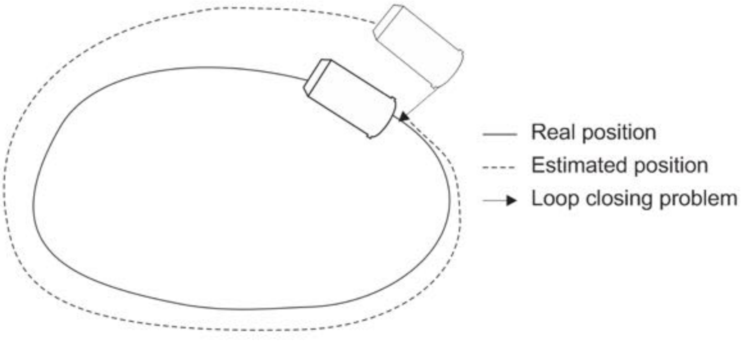


Figure 4.6: Loop closing in SLAM

4.3.2 FastSLAM

FastSLAM is an alternative approach to SLAM introduced by Montemerlo [126], in which the probability distribution for SLAM (Eq. 4.1) is divided into factors through a Rao-Blackwellization method and solved using PF and EKF. PF establishes the localization as a set of particles, each particle a possible pose, which it is weighted through time, given the observations [88].

A graphical model of SLAM is represented in Fig. 4.7. It shows the changes of poses x_{t-1} to x_t given the control signal u_t, the observations z_{t-1}, the pose x_t and the landmarks m_i. In this representation, the arrows show direct dependencies. Since there is no direct relationship between poses and landmarks, a conditional independence is established. This conditional independence between landmarks and poses is exploited by Rao-Blackwellization to model the robot path sampling and computing the landmarks once the poses are known [87, 88].

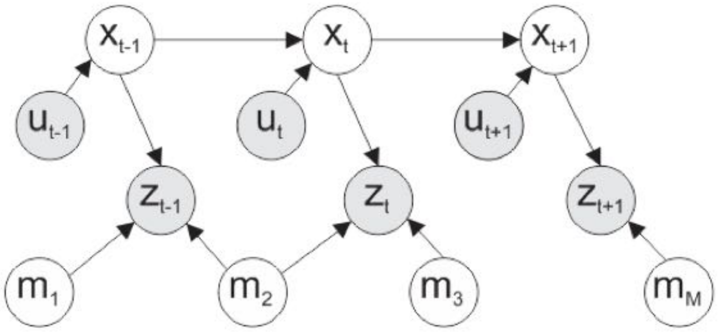


Figure 4.7: SLAM Graphical model
Redrawn from [122]

Particle Filter for Localization

PF works with multiple assumptions that represent possible poses (particles). Each of them is associated with a weight that denotes how accurate the estimated particle (posterior) embodies the new incoming data from the sensors (observations).

$$x = \left\{ \left\langle x^{[i]}, w^{[i]} \right\rangle \right\}_{i=1,\ldots,N} \tag{4.10}$$

Where $x^{[i]}$ is a sample or state hypothesis with an associated weight $w^{[i]}$, and N is the number of samples. The weights are normalized in order that the sum is equal to 1. Therefore, the posterior state is represented by [127]:

$$p\left(x\right) = \sum_{i=1}^{N} w^{[i]} \delta_{x[i]}(x) \tag{4.11}$$

The weights are chosen through the *Importance Sampling Principle,* which established that another probability density g (*Importance Density*) can be used from which is easier to draw samples and evaluate $g(x)$ [128], to define the weights as follows:

$$w_t^{[i]} = \frac{target(x_t^{[i]})}{ImportanceDensity(x_t^{[i]})} \tag{4.12}$$

In localization, the importance density is given by the motion model and the weights are proportional to the *observation model.* Weights represent how likely is the estimated particle (posterior) compared to the *'reality'* (observations) [123].

$$x_t^{[i]}\ p(x_t|x_{t-1},u_t) \tag{4.13}$$

$$w_t^{[i]} = \frac{target(x_t^{[i]})}{ImportanceDensity(x_t^{[i]})} \propto p(z_t|x_{t,}m) \tag{4.14}$$

The implementation of the PF for localization begins with all particles having the same weight. As the iterations are performed and the vehicle perceives new observations the samples change weights resulting in an accurate localization area. It also allows multimodal estimations which are not possible with EKF. This means, at a particular time different probable locations can be handled in the model [88, 129].

The resampling or sampling with replacements process is used to reinforce the more likely areas due to the weight and eliminate samples with low probability to maintain the number of samples. It also helps to filter the noise created at the time of propagation [87, 88, 130, 131].

Rao-Backwellized PF for SLAM

PF reduces its performance when working in a high-dimensional space such as SLAM where the state vector includes the robot state vector and the map landmarks matrix. To reduce the complexity for solving the whole SLAM problem with PF, dependencies in the state vector are exploited from the concept that the landmarks can be computed after a given pose estimation. From each particle (each pose hypothesis) a map can be calculated and then weighted as in a PF method.

Based on the conditional independence between features and poses (Fig 4.7), the SLAM probabilistic model is then decomposed in 4.15 [132] where the first term corresponds to the path posterior and the second to the map posterior.

$$p\left(x_{0:t}, m_{1:M}\left|z_{1:t}, u_{1:t}\right.\right) = p\left(x_{0:t}\left|z_{1:t}, u_{1:t}\right.\right) p\left(m_{1:M}\left|x_{0:t}, z_{1:t}\right.\right) \tag{4.15}$$

Then, since landmarks are conditionally independent given the poses, the map posterior is factorized in a product of conditional landmarks. Therefore, the path posterior is solved through PF, and the products through low-dimensional EKFs [132].

$$p\left(x_{0:t}, m_{1:M}\left|z_{1:t}, u_{1:t}\right.\right) = p\left(x_{0:t}\left|z_{1:t}, u_{1:t}\right.\right) \prod_{i=1}^{M} p\left(m_i\left|x_{0:t}, z_{1:t}\right.\right) \tag{4.16}$$

A simplified block diagram is presented in Fig. 4.8. Each particle represents a different path, and for each path posterior, a map is computed. Particles are then weighted based on the likeability between the observation model for each computed map. Then a map is updated through the computation of EKFs.

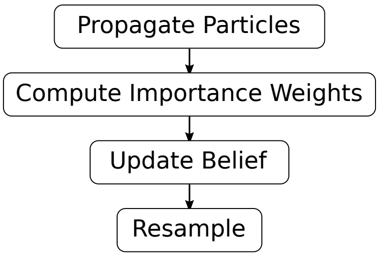


Figure 4.8: Particle Filter simplified block diagram

FastSLAM allows multimodal estimations which are not possible with EKF (it can maintain different possible locations at a time). The fixed number of samples reduces the computation cost. On the other hand, a drawback of this method is that it loses information by resampling, therefore, in some applications, this process has to be enhanced [133].

4.3.3 Graph SLAM / Smoothing

Graph SLAM is an intuitive method presented in [134] where the trajectory of the robot and the observations are denoted as nodes and edges that represent constraints. The estimation of the state vector is done by optimizing the arrangement of nodes through the computation of nodes and edges. The method explained here is known as offline SLAM since the estimation is calculated through the total of previous poses $x_{1:k}$ [89].

The probabilistic representation is [84]:

$$P(x_{1:k}, m|Z_{0:k}, U_{0:k}, x_0) \tag{4.17}$$

Sensor measurements are abstracted to “virtual measurements” z_{ij} which is a transformation from the observations gathered from the nodes x_i and x_j to maximize the overlap, the matrix Ω_{ij} (information matrix) represents the uncertainties, as constraints from the measurement z_{ij}. On the other hand, $\widehat{z}_{ij}$ (x_j seen from x_i) is an expected observation obtained by the relative vector between the nodes x_i and x_j. Then the error, $e_{ij}(x_i, x_j)$, is the difference between both (Fig. 4.9).

Finally, the goal of the approach is to build the graph, then find a node configuration that minimizes the error e_{ij} given the constraints [89].

$$x^* = \underset{x}{argmin} F(x) \tag{4.18}$$

Where:

$$F(x) = \sum_{\langle i,j \rangle \in C} F_{ij} = \sum_{\langle i,j \rangle \in C} e_{ij}^T \Omega_{ij} e_{ij} \tag{4.19}$$

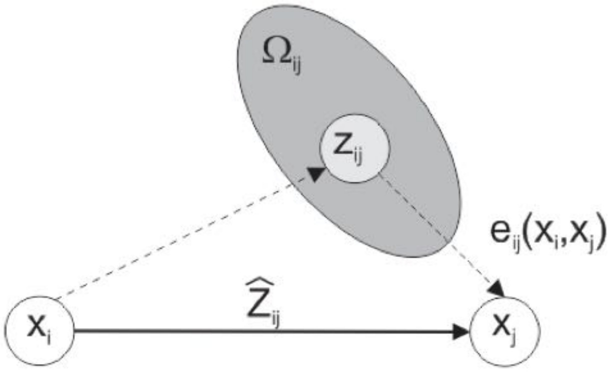


Figure 4.9: GraphSLAM model. Each node is a pose. Edges represent observation constraints from landmarks and how x_j is seen from x_i
Redrawn with permission from [89]

In a recursive algorithm, GraphSLAM can be represented as an iteration between building the graph (front-end) and then optimize the graph (back – end). The graph is constructed from the data collected as nodes & edges (or poses & constraints) and then is optimized finding the best for all of them Fig. 4.10.

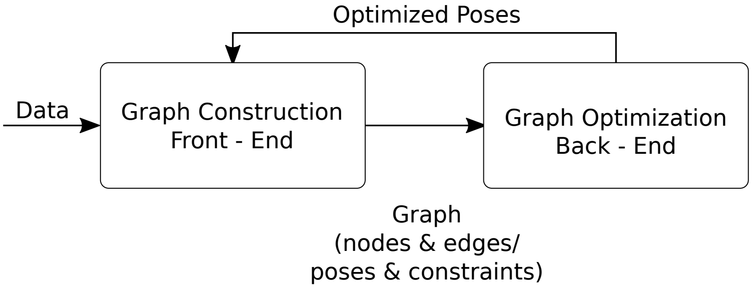


Figure 4.10: Overall block diagram for a GraphSLAM system

GraphSLAM is becoming popular in SLAM, and there are different state-of-the-art approaches to improve the error minimization [135]. Some code examples and practical algorithms can be found in [89].

4.3.4 SLAM in Underwater Applications

There are many applications for SLAM in underwater environments. The application defines the selection of perception sensors and onboard sensors for navigation. Then, the SLAM problem can be stated from different approaches depending on the number of landmarks, the area of coverage, computational requirements, flexibility, etc. [136]. Important aspects of the SLAM approaches reviewed in these documents are compared in Table 4.1.

Several extensions of the approaches presented have been developed and applied as shown in Table 4.2, where a summary of selected marine applications is presented. It reveals that structured environments are mostly approached using MSIS, FLS, and video cameras (in that order of importance). To apply SLAM in these scenarios, complex walls, borders, and points are used as features in surveillance and inspection applications.

Unstructured environments are mostly referred to seafloor applications where the use of SSS, Video cameras and FLS excels. Most of the applications involve the use of DVLs and IMUs for localization estimation.

Perception sensors are used arbitrarily with different SLAM approaches according to the application. Depending on the output of the perception sensor, different feature extractors and scan-matching methods are used.

Table 4.1: SLAM approaches overview

Concept	KF	PF	GraphSLAM
Computational Effort Complexity	m^2 m: # features	N*log(m) n: # samples m: # of features	Linear to the number of constraints Linear to number of nodes
Assumed Distribution	Gaussian	Pose: Arbitrary landmarks Distribution: Gaussian	Gaussian or different cross function
Linearization	KF all linear EKF one linearization	Motion does not need linearization	Re–linearize in every direction
Landmarks handled	Hundreds	Thousands	Thousands
Flexibility	Medium	High	Higher
Large Scale	Poor	High	Depends on sparsification

In [107], a two-stage EKF algorithm for partially structured environments, SLAM is used with a DVL and an MSIS. Due to the distorted images collected from the MSIS while the mobile is moving, a first EKF is used to estimate the trajectory and then correct the MSIS images. The second stage is an EKF-SLAM using motion estimation and undistorted images once data association is performed for lines from planar structures such as dams, harbor or platforms [108].

In [110] a SLAM system is used in an AUV for mine counter measurement and localization. An apriori map from a SSS and a FLS as a perception sensor are used. The graph is initialized by pose node from a GPS. A nonlinear least squares optimization is performed (variation of GraphSLAM) to the dead-reckoning (DVL and IMU) sensor and sonar images. In [137] an approach of Graph SLAM is proposed using a set of membership to perform SLAM in a non-linear environment from seamarks located through a SSS.

A visual SLAM using monocular video images through a novel saliency method using local and global saliency for feature detection in hull inspection is shown in [116, 138]. A Viewpoint Augmented Navigation (VAN) framework using a large number of features extracted from stereo images for optimizing visual loop closures is presented in [139]. An application of a VAN for a monocular camera is shown in [140] (Eustice, Pizarro, & Singh, 2008) in low overlapping images and unstructured environments.

A multi-pencil sonar is used in [130] for SLAM. The sensor offers low-resolution range observations suitable for Rao-Backwellized applications. A multibeam sensor is used in a Bathymetric-distributed Particle SLAM (BPSLAM) in [141]. Previous low-resolution bathymetric maps are used to aid the navigation, while a high-resolution mapping is performed by applying an adapted distributed particle mapping [142]. A robust real-time localization for large maps method is presented in [106] through the use of grids.

In [143] a MSIS is used through a modified FastSLAM 2.0 method in a navigation system GPS, AHRS, Compass, DVL are used for localization. Height problems are not solved entirely using MSIS since features are localized in a 2D space.

4.4 SLAM Implementation in Underwater Environments

4.4.1 SLAM Implementation

The three SLAM frameworks discussed before are implemented for an underwater application based on adaptations of the source codes of Tim Bailey for EKF SLAM and FastSLAM [144], and Salim Chedrawi for GraphSLAM [145]. The objective of the simulation is to show the applicability of the three frameworks for the same collected data from a simulation environment. Its worth noticing that the implementations are simple versions of more elaborated developments as those shown in Table 4.2. The test does not include loop closure, submapping or any relocalization algorithm in order to show the basics of the estimations evolutions for the methods fundamental principles. The simulation time is limited to focus on the performance of the algorithms in every step when new information from the sensors is gathered.

An AUV equipped with a camera and a displacement sensor is simulated in UWSim [71], an

Table 4.2: Summary of selected works on underwater SLAM

Ref	**On-Board Navigation Sensor**	**Percept. Sensor**	**SLAM**	**FE or SMM**	**Scenario**	**Large Map**	**Close Loop**	**Application**
[114]	IMU, DVL, GPS	Mono camera	-	SIFT/SURF	Unstructured	-	Possible	Sea floor based Navigation
[146]	DVL, Gyroscope	FLS	EKF	Points and lines	Unstructured Shallow Water	-	-	Surveillance
[105, 108]	DVL, IMU	MSIS	EKF	Hough-Based	Structured	-	-	Channel Navigation
[107]	DVL	MSIS	EKF	Line Match	Partially Structured	-	Possible	Harbor, Dams Navigation
[147]	GPS, DVL, Acoustic transducer	SSS	EKF	Salient features	Unstructured	Sub-mapping	Yes	Sea floor Navigation
[112]	DVL, Compass	SSS	EKF – Augmented State	Filters	Unstructured	Possible	Multi pass (2 – 3 times)	Sea floor Navigation
[148]	DVL	FLS	EKF - Exactly Sparse Extended Information Filter (ESEIF)	Manual	Structured	-	-	Shipp Hull Inspection
[149]	DVL, IMU	MSIS	EKF - Robocentric IE KF	spIC	Structured	Possible – Sub-mapping	Yes	Tank, Harbor Navigation
[150]	IMU Gyroscope Compass	MSIS	Fast-SLAM	Filter	Structured	Possible	Yes	Channel Navigation
[151]	None	MSIS	Fast-SLAM	Filter, Support Vector Machine	Structured	-	-	Wall sufficiently complex Navigation
[109]	DVL, Accelerometer, Depth	FLS	Graph-SLAM	Pair-wise (frames), Threshold, clustering	Structured	-	-	Surveillance
[110]	Altimeter, GPS, Compass	FLS	Graph-SLAM	Filters Blue View	Unstructured	Possible	Yes	Sea floor Navigation
[135]	IMU	MSIS	Graph-SLAM	uspIC	Simulation	-	Yes	Simulated Channel Navigation
[152]	DVL, IMU	Mono Camera	Graph-SLAM	Template Matching	Structured Artificial Landmarks	-	-	Experimental

continued on next page

continued from previous page

Ref	On-Board Navigation Sensor	Percept. Sensor	SLAM	FE or SMM	Scenario	Large Map	Close Loop	Application
[153]	DVL, Gyro, Depth	Stereo Camera – laser line	Graph-SLAM	Scale Invariant Feature Transform (SIFT)	Deep water	Sub-mapping	Yes	Bathymetry
[154]	DVL, IMU	MSIS	Graph-SLAM Augmented State EKF (ASEKF)	Iterative Closest Point (ICP) - MSISpIC	Structured, Possibly Unstructured	-	Yes	Channel Navigation
[141, 142]	DVL, Gyro, Compass, Depth	MES	Grid Mapping, RB		Previous Map	Grids	-	Bathymetry with previous map
[155]	Doppler INS	Side Scan Sonar	EKF	Image processing and template match	Unstructured	Yes	Yes	Autonomous navigation
[156]	Visual Odometry	Stereo Camera	EKF, Graph-SLAM	-	Unstructured. Seabed	Yes	Yes	Navigation / Seafloor 3D model
[157]	Visual Odometry	Mono/Stereo Camera	RT-SLAM, EKF	Corner detection	Structured environment/ pool	-	-	Robot Inspection
[158]	IMU, DVL	Camera	Augmented State Kalman Filter (ASKF)	Selective images. SIFT	Underwater structures. Hulls	Yes	Yes	Structures inspection
[159]	Depth, Visual Odometry	Stereo Camera	GraphSLAM	Oriented FAST and Rotated BRIEF (ORB)	Unstructured. Seabed: Seagrass, sandbanks	Yes	Yes	Seafloor Navigation

underwater simulation environmnet for robots vehicles with Robot Operating System (ROS) integration. The robot dives into a pool where simple blocks are placed on the bottom as landmarks. The purpose of the setup is to apply SLAM to create a map of the blocks and localize the robot in it (Fig. 4.11).

The robot movement is limited to a fixed depth and two degrees of freedom (surge and yaw). Two thrusters for the surge are powered to obtain a linear motion while the blocks of the bottom are recognized through image processing. Samples are taken every 0.6 seconds in which the real position of the robot, displacement sensors, and blocks observations are logged.

During each observation, an image processing algorithm is run to observe the block from the known current position of the robot. A Nearest Neighbor (NN) [160] algorithm is implemented to assign a tag to each block observed. The noise considered in the prediction from the motion are modeled as Gaussians with a standard deviation of σ=0.03m in X and Y. The noise in the observation model from the landmarks seen by the camera are modeled as Gaussians with standard deviation of σ=0.11m and σ=0.785 rad, respectively for range and bearing. A 2D simple model is used for prediction and observation.

Prediction model for the robot pose:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = X_k = \begin{bmatrix} x_{k-1} + \Delta x \\ y_{k-1} + \Delta y \\ atan2(\Delta x, \Delta y) \end{bmatrix} \tag{4.20}$$

Observation model:

For each re-observed landmark its observation model is:

$$z = \begin{bmatrix} \sqrt{(m_x - x_k)^2 + (m_y - y_k)^2} \\ atan2\left(m_y - y_k, m_x - x_k\right) - \theta_k \end{bmatrix} \tag{4.21}$$

Augment model:

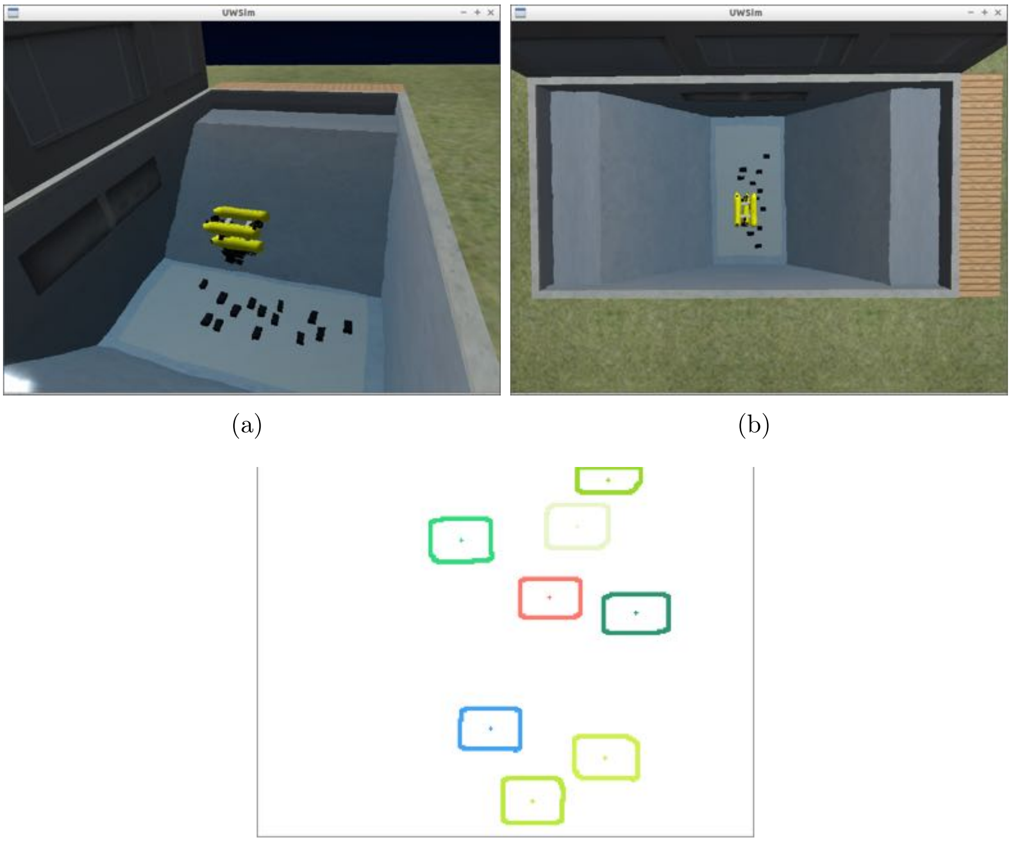
For each new landmark given a relative range r and bearing b is recorded, its absolute position is calculated:

$$m_k = \begin{bmatrix} x_k + r.\cos(\theta_k + b) \\ y_k + r.\sin(\theta_k + b) \end{bmatrix} \tag{4.22}$$

The prediction model and the augmented model are used to estimate the path and the landmarks of the logged data without performing any SLAM algorithm as a baseline, the final landmark's position is obtained averaging all the observations (Fig. 4.12). The blue color indicates the real path and landmark locations and, the red color, the estimated. The observation circles represent the different estimations of the landmarks at every step of the simulation, having a different color per landmark. The path and landmarks exhibit a maximum error of 1m and 0.7m respectively. It is also helpful as an overview to display how dispersed the data is (observations). The black dashed lines link the real landmarks with their estimations.

4.4.2 Results

The data is processed using the SLAM approaches, and the results are shown in Fig. 4.13. In EKF, pink ellipses represent the uncertainties. Additionally, in FastSLAM the uncertainty ellipses also represent uncertainties and the dots, other particles. It worth mentioning that the landmarks and path drawn for FastSLAM are for the highest weighted particle in the end. In the case of GraphSLAM, an offline approach is implemented, this means that all the logged



(a) (b)

(c)

Figure 4.11: Simulation Setup. (a) and (b) Underwater robot diving in a pool with fixed landmarks. (c) Processed images with data association, each color represents a different tagged landmark.

data is processed at once for estimating the last position and path.

Fig. 4.14 and Fig. 4.15 show the mean squared error (MSE) for the position at every step and the final landmarks respectively. The 'without SLAM' path and landmark errors start to increase along with the number of steps and with the further landmarks given that the errors from the sensors are being accumulated without any feedback.

Conversely, the SLAM approaches present a better response over time. The position errors are kept in range, and the landmark errors tend to be constant since the position is being improved.

The three approaches handle the localization and mapping in a better way than a simple averaging algorithm. While it was expected that the error grows over time giving the accumulated uncertainties for the case without SLAM, the SLAM approaches prevent the error from a constant raise. The best result is obtained by GraphSLAM, which is applied as an offline algorithm, with

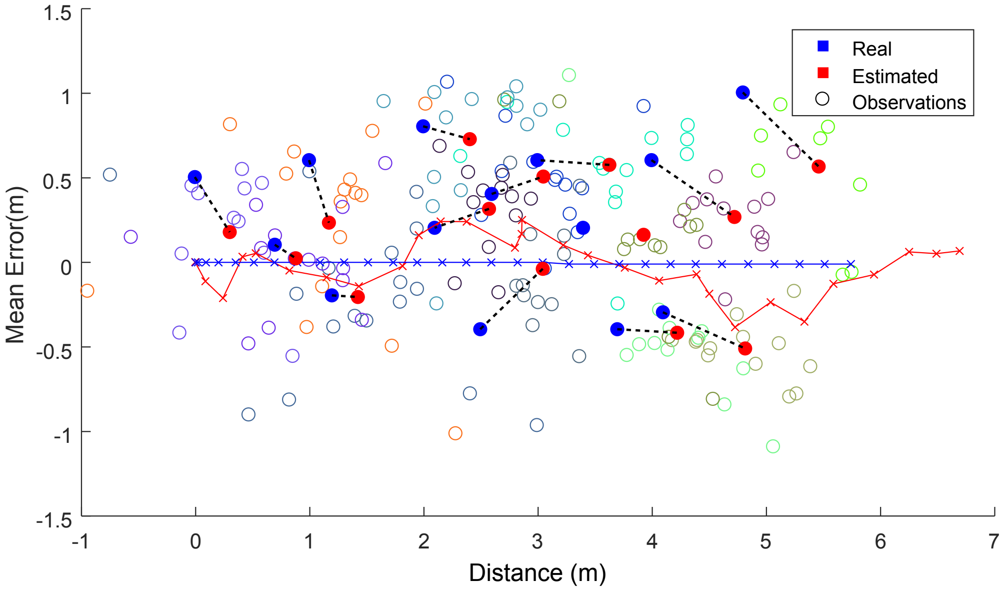


Figure 4.12: Estimated path and landmarks without SLAM, lines associate real and estimated landmark positions

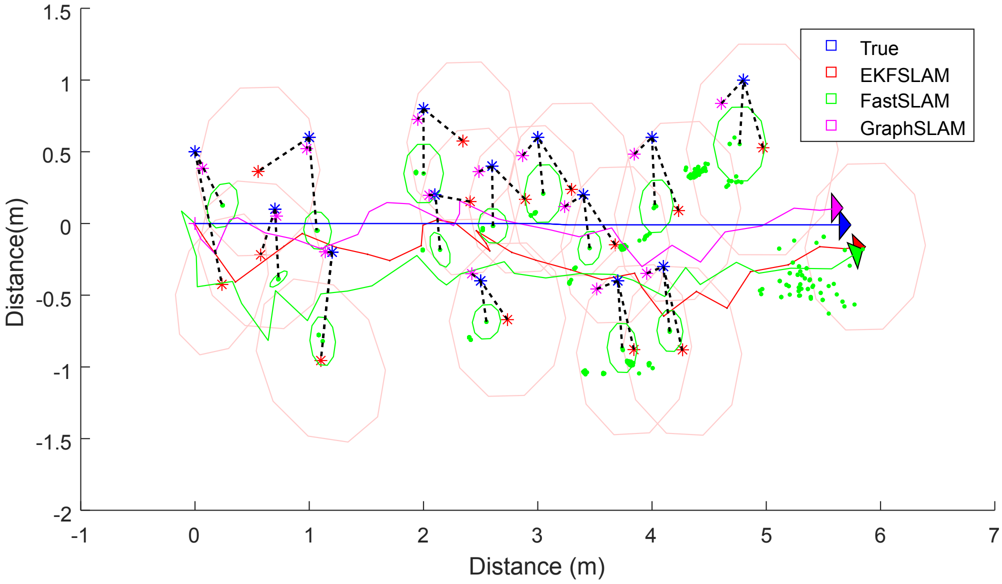


Figure 4.13: Results of SLAM: path and landmarks

a final Mean Squared Error (MSE) of less than 0.2m compared to without SLAM with 1m at the last step. In average the MSE of GraphSLAM is 0.2m compared to 0.5m without SLAM.

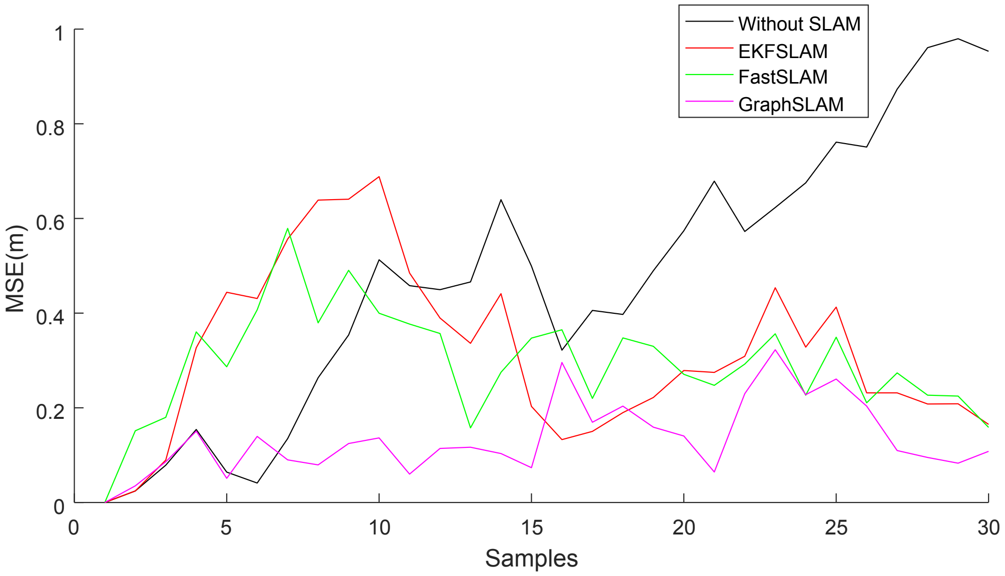


Figure 4.14: Position MSE at every step

4.5 Chapter Summary

We have reviewed the fundamentals approaches for most of the state-of-the-art SLAM algorithms. The simulation shows how they process the information gathered from the localization and exteroceptive sensors to improve the estimation of the pose and map generated. Current algorithms improve feature extraction, data association, map generation, sensor fusion, etc. for different scenarios and applications.

Recent development in onboard instrumentation for localization and data fusion contribute to an accurate localization estimation. On the other hand, a variety of underwater sensors for the perception of the environment have been reviewed in their applications for SLAM for different approaches and scenarios.

The sensor selection relies on the application of two primary scenarios: structured and unstructured. In the first case, the structures are exploited to generate the map: MSIS, FLS, and video camera are mostly used. In the second case, features on the seabed are taken as landmarks: SSS, video cameras, and FLS are used in this case. The selection of the SLAM approach mostly depends on the information gathered and processed from the perception sensors, the size of the area of work and the processing capacity of the controller.

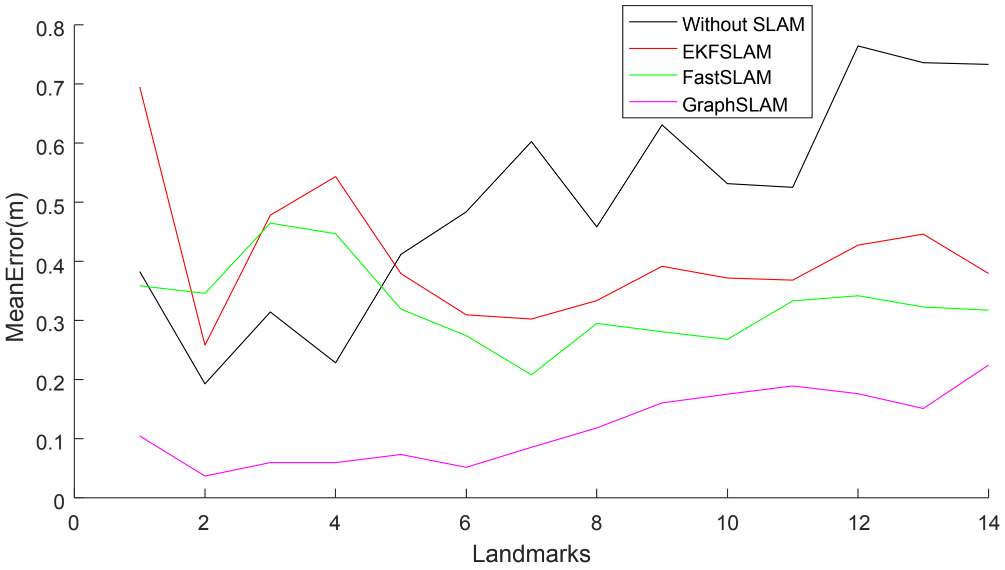


Figure 4.15: Landmark MSE at the final step

Applications for developments of SLAM in underwater scenarios include seafloor navigation, structures monitoring, surveillance and navigation in confined environments. Different methods have been applied since the late 90's/ early 2000's and are still being developed in the present (Table 4.2). The applications of underwater SLAM suits industry demand from oil & gas structures inspection, environmental monitoring, hydrography, and search&recovery.

Chapter 5

Interest Point Detectors and Descriptors for Underwater Visual SLAM

Modern visual SLAM (vSLAM) algorithms take advantage of computer vision developments in image processing and in interest point detectors. They are used to create maps and estimate the trajectory of a robot through the images gathered by a camera mounted on it. Different feature detectors have been evaluated for this purpose in air and ground environments, but not extensively for underwater where light and suspended particles in this medium alter considerably the images captured. In this chapterwe present a comprehensive survey of the application of interest points detectors and descriptors in a variety of underwater scenes and conditions. We classify scenarios based on the alterations and evaluate their performance towards vSLAM.

5.1 Introduction

It is essential to know the position of underwater robots and to obtain maps of the surrounding environment for a variety of robot tasks, from monitoring, geo-referencing gathered data to autonomous navigation and exploration. SLAM offers a framework to incrementally build a map while a robot moves through an unknown area and to use that map to localize the robot simultaneously. A typical implementation of SLAM in the underwater environment involves the use of dead-reckoning, acoustic sensors and cameras [85]. In the last few years, the use of cameras as the primary sensor for SLAM has increased. This branch of SLAM is also referred to as visual SLAM (vSLAM) which mainly focuses on estimating the pose of the camera from partially overlapping images from different viewpoints and creates a map of images or a cloud of points. Visual SLAM can be categorized based on how the images are processed in direct algorithms, where complete image intensities are processed, and feature-based, where only certain key-points of the image are computed [161].

A fundamental part of feature-based SLAM is data association which allows extracted features from images as key-points to be recognized when re-observed in consecutive images as well as in loop-closing. To achieve this, features are extracted using an interest point detector, and then described including local information from the neighbors of the point through a feature descriptor. The descriptor is a vector which assigns a distinctive identity to the feature to be recognizable [162].

In computer vision there are several feature detectors and descriptors which have been evaluated in terms of scale invariance, viewpoint changes (including rotation) and variations in illumination [162–167], as well as their application to vSLAM [158, 162, 168, 169]. There are successful implementations of vSLAM for underwater robots such as in [114, 116, 157, 170] which relies on Speeded-Up Robust Features (SURF), Scale Invariant Feature Transform (SIFT) feature detectors and other methods to extract regions of interests. To the best of the authors' knowledge, there is not extensive documentation which analyses feature detectors and descriptors for underwater environments. This might be related to the higher number of applications of point detectors and descriptors in indoors environments and, in images captured by ground or air robots compared to underwater environments which present images with dynamic illumination, blurriness, turbidity; and there are fewer targets from which features can be extracted, mostly limited to man-made structures, animals or the seafloor, which can be affected by the currents such as in the cases of sand patches and algae.

Underwater images are subject to alterations to the light and characteristics of the medium resulting in blurry, hazy and tinted images [171, 172]. This presents challenges to the performance of feature detectors towards vSLAM. Therefore, we propose a characterization of underwater scenarios based on a variety of datasets in different conditions and evaluate the response of common feature detectors and descriptors such as SIFT, SURF, as well as newer approaches such as Oriented FAST and Rotated BRIEF (ORB), Binary Robust Invariant Scalable Keypoints (BRISK) and AKAZE. We include some processed datasets through enhancing algorithms. Additionally, we evaluate the descriptors performance in matching consecutive images. Finally, we compare the computation time for the features detection and matching.

In the second part of this Section, a selection of related works to feature detectors and their evaluation in vSLAM is presented. Section 5.2 presents a brief overview of selected feature detectors with their corresponding descriptor. Section 5.3 presents the alterations found in underwater images as well as a brief description of enhancement algorithms. The evaluation methodology and the results are presented in Section 5.4 and Section 5.5. Finally, the results are discussed in Section 5.6 together with the conclusions.

5.1.1 Related Work

A recent survey specifically on visual monocular SLAM was presented by Younes et al. [161]. They outlined a general guideline of a monocular keypoints SLAM system defining seven components: "visual initialization, data association, pose optimization, topological/metric map generation (map expansion), bundle adjustment/pose-graph optimization/map maintenance, failure recovery and loop closure". In [173], typical modern SLAM architectures are abstracted to front-end and back-end components. The first one extracts sensors data and pre-processes it to be handled by the back-end to infer a consistent map and pose estimation. In this representation, the data association process fits in the front-end leaving the other modules to the back-end Fig. 5.1.

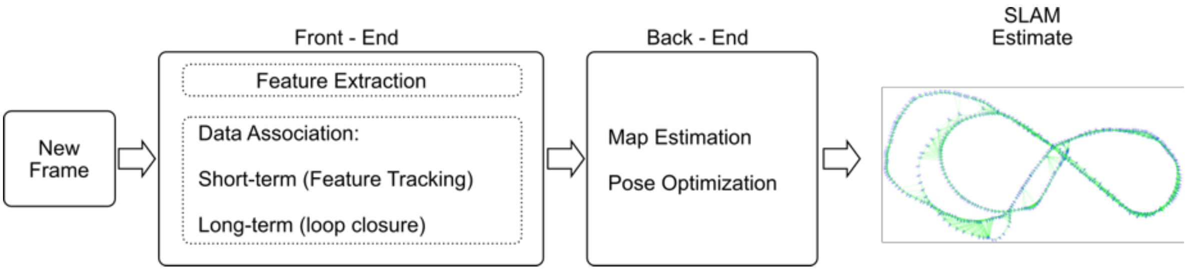


Figure 5.1: Simplified vSLAM architecture

For feature-based vSLAM, the front-end involves the detection of interest points, the creation of descriptors, and the data association performed by matching features from the current frame with previous frames.

5.1.2 Feature Detectors in Visual SLAM

Visual SLAM approaches have been evaluated for indoor and outdoor applications over benchmark datasets. In [174] ORB-SLAM, LSD-SLAM, L-SLAM and OpenRatSLAM algorithms are briefly described and assessed. ORB-SLAM shows good results for different environments presenting the smallest errors when compared to LSD-SLAM and RAT-SLAM. The authors also pointed out the need of manual post-processing to reduce the error since the maps and trajectories need post-scaling to fit proper dimensions.

In [175] an experimental evaluation of the algorithms was performed for different datasets collected on land, aerial and underwater vehicles. They found, again, a good performance by

ORB-SLAM and Parallel Tracking and Mapping (PTAM) for the majority of scenarios. Finally, another evaluation was performed in [169] having similar results with three different feature detectors: Harris, Kanade-Lucas Tracker (KLT) and SIFT.

5.1.3 Feature Detectors Evaluation

Several feature detectors and descriptors have been evaluated in the past regarding correct matching against image alterations. In [176] the SIFT descriptor was evaluated with ground truth showing robustness against rotation, scale, viewpoint changes, image blur and light change. They define three ratios, first used in [177], to measure the performance of the measurements.

$$Recall = \frac{\#Correctmatches}{\#Correspondences} \tag{5.1}$$

$$1 - Precision = \frac{\#Falsematches}{\#Matches} \tag{5.2}$$

$$Recall = \frac{\#Correctmatches}{\#Detectedfeatures} \tag{5.3}$$

Johansson et al. use the same performance ratios to evaluate more detectors and descriptors; and combinations (detectors/descriptors). They include SURF, ORB, BRISK, Fast Retina Keypoint (FREAK) finding the combination SURF/SURF and ORB/BRISK robust against geometric and photometric transformations [166]. Similarly, Gil et al. show SURF and Gradient Location and Orientation Histogram (GLOH) (a SIFT like descriptor) suitable for a vSLAM application [162].

Other evaluations are performed for custom applications such as for tracking objects [165] and vision-based localization [178]. They introduced the Accelerated-KAZE (AKAZE) detector/descriptor to the review and compared the computing time among them. In [178] they include Compute Unified Device Architecture (CUDA) implementations of AKAZE and SIFT being the fastest two in extracting, detecting and matching, followed by ORB and SURF. SIFT appeared as the slowest followed by AKAZE and BRISK. Additionally, they add repeatability, precision and accuracy as comparison criteria.

5.2 Selected Feature Detectors and Descriptors

Based on the performance of features extractors in the literature discussed in Section 5.1 we selected SIFT [179], SURF [180], ORB [181], AKAZE [182] and BRISK [183] which are robust and have been successfully evaluated for indoor and outdoor environments in [178, 184]. In

Table 5.1 the characteristics of the detectors and descriptors are presented, as well as some parameters based on their OpenCV implementation.

Table 5.1: Detectors / Descriptors characteristics and parameters

Detector / Descriptor	**Features to Detect**	**Size of Descriptor**	**Parameters**
SIFT	Blobs	128 Bytes	Contrast Theshold, Sigma
SURF	Blobs	128 Float	Hessian Threshold
ORB	Corners	32 Bytes	Fast Threshold, Max Features
BRISK	Corners	64 Bytes	Brisk_threshold
AKAZE	Blobs	61 Bytes	AKAZE_threshold

Scale Invariant Feature Transform (SIFT)

The SIFT algorithm follows two main stages in the detection part: (1) Scale-space extrema detection, where Difference of Gaussian (DoG) is applied to identify keypoint invariants to scale changes, then a local extrema check with adjacent pixels is performed; (2) keypoint localization, which rejects low contrast keypoints and then eliminates non-edge points based on Hessian matrix.

For building the descriptor the algorithm follows two further steps: (3) orientation assignment, which forms orientation histograms from local gradients to determine the dominant direction of the keypoint; (4) keypoint descriptor, where the proper vector is constructed based on the course of the keypoints and local areas around them, and finally the descriptors are normalized to improve light invariance [179, 185].

Speeded-Up Robust Features (SURF)

SURF follows a similar idea as SIFT, it was developed by Bay et al. [180] as a faster and robust alternative to previous extractors. It uses integral images [186] and simplified filter kernels compared to SIFT through a Fast-Hessian detector based on 2D Haar wavelet response.

The descriptor combines local gradient information, like SIFT, 2D Haar wavelet response to local areas and windows around they keypoints to approximate the gradients.

Oriented FAST and Rotated BRIEF (ORB)

ORB is based on Features from Accelerated Segment Test (FAST) and Rotated BRIEF. It creates a pyramid of blurred and subsample versions of the image which are then divided into cells and FAST is computed. Then the cells are subdivided to contain one corner per cell or the maximum number of cells allowed by a parameter of the algorithm, disregarding the features with low score per cell.

The ORB descriptor modifies the FAST extractor adding an orientation component through first-order moments in a local patch. Then the Binary Robust Independent Elementary Features (BRIEF) descriptor is computed on a rotated patch. It reduces the descriptor vectors such as in SIFT and SURF to binary vectors [181].

Binary Robust Invariant Scalable Keypoints (BRISK)

BRISK is based on the FAST detector, it extracts features from the image and different scales of it. For the descriptor, it uses a concentric rings sampling pattern to retrieve the gray values of their neighbors and process local intensity gradients to obtain the direction of the keypoint. Then it forms the binary descriptor comparing the intensity between pairs from the pattern [183].

Accelerated-KAZE (AKAZE)

AKAZE focuses on multi-scale feature detection exploiting non-linear scale spaces. It is computationally efficient taking advantage of Fast Explicit Diffusion. It applies the Hessian determinant to the scaled images and performs a search of the maxima response in spatial location.

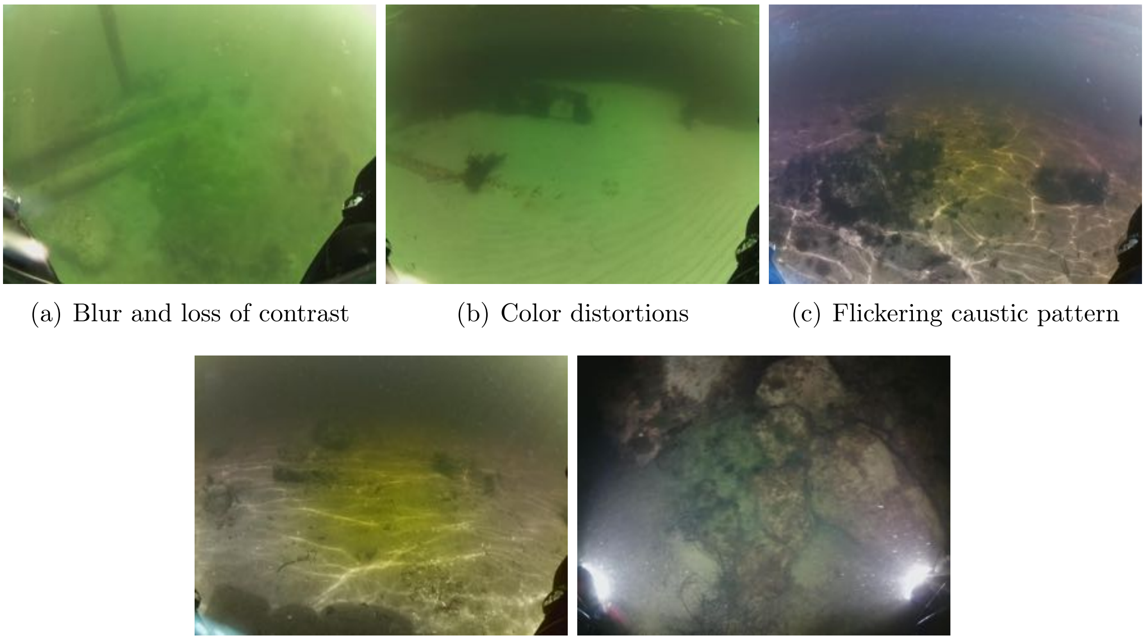
Alcantarilla et al. proposed a Modified-Local Difference Binary (M-LDB), that exploits gradient and intensity from the extractor stage, as a descriptor. It is based on BRIEF performing over the average of areas instead of pixels. It includes intensity values, and the orientation of the keypoint is similar to KAZE [182].

5.3 Underwater Monocular Images

Images captured in underwater scenarios are altered in every aspect due to the changes in radiant energy when traveling through water rather than air. Light gets scattered by suspended tiny particles in the water (quartz sand, clay mineral, plankton, etc.) and it is also absorbed by the water itself causing blur and loss of contrast (Fig. 5.2 (a)) [187, 188]. The energy absorption varies with wavelengths and types of water (i.e. sea, fresh and variations in its composition), generating perceived color distortions (Fig. 5.2 (b)) at different distances and types of water. Additionally, changes in perception of size and distance also occur in underwater scenarios and are caused by the light refraction as it passes from air to water [171, 172].

Sunlight flickers (caustic waves) are observed in very shallow water which are formed by trespassing a wavy water layer [189]. These lighting variations generate flickering caustic patterns (Fig. 5.2 (c)), which can be seen as random thin bright traces and non-uniform illumination, which are observable as brighter small patches (Fig. 5.2 (d)) [114].

Artificial light sources are used when gathering images at night or in murky water to increase the lightness of the scene. The source is usually located near the camera and the light is reflected by particles in the medium yielding the back-scatter component (Fig. 5.2 (e)) [190, 191].



(a) Blur and loss of contrast

(b) Color distortions

(c) Flickering caustic pattern

(d) Non-uniform illumination

(e) Back-scattering

Figure 5.2: Lighting effects on underwater images

5.3.1 Underwater Image Enhancement

There are several approaches of image processing to enhance underwater images regarding the lighting effects presented before. In [191], Wang lists around 25 different algorithms for underwater image enhancement and restoration. The author organized them in four categories, having ‘Histogram and Contrast Ratio’, which mainly enhances the contrast; ‘Retinex Model’, with good results in low contrast and non-uniform illumination; ‘Filtering and Transformations’, which also enhances non-uniform illuminated images, corrects the image tone, reduces noise of bright spots and improves contrast; and ‘Comprehensive’, which enhances and restores colors in the images.

Other methods developed mainly to diminish the effect of sunlight flickering such as the works presented in [189, 192–194]. Additionally, the algorithms presented in [190, 195] enhance underwater images with respect to the back-scattering problem. 'Dehaze' algorithms have also been used to overcome the light scattering problem in air [196–198] and in water [199].

External hardware have been used for mitigating the lighting problems when gathering underwater images. Treibitz et al. placed polarizers on the light source and the camera to achieve back-scatter reduction [200, 201]. In [202], a barrier filter was used in front of the camera for the same purpose.

5.4 Evaluation Framework

In this Section we present the evaluation framework followed, based on the literature described in Section 5.1 from [114, 162, 163, 166]. A quantitative and qualitative analysis is performed to evaluate the performance of feature detectors and descriptors applied to underwater images toward their application to vSLAM.

We also include processed images from the datasets through an enhancing underwater image by fusion [203] and backscatter removal to enhance the visibility of underwater objects [195].

The profiles have been tuned manually to expose features proportionally to the limit. One is set to obtain around 1000 features ($Profile_{1k}$) and the other to achieve a higher value, limited to 10000 features ($Profile_{10k}$). The profiles are based on the threshold of the extractors and the number of maximum parameters (Table 5.2). The other parameters are left to the default values of the OpenCV implementation of the algorithms.

Table 5.2: Profile parameters changes

	$Profile_{1k}$	$Profile_{10k}$
Max Features	1000	10000
Threshold		
SIFT Contrast	0.01	0.008
SURF Hessian	60	8
ORB Edge	32	8
BRISK	10	7
AKAZE	0.0005	0.0001

5.4.1 Detectable Features in Underwater Images

We describe a selected number of underwater datasets based on the challenges presented in 5.3 and evaluate different feature detectors on them to determine distinctive image features in underwater scenarios. Processed images are also included to observe how the enhancement performs when features are extracted.

The two features extractors profiles are included in the analysis. Quantitatively, the number of features extracted is given. A qualitative description of the detected features in different scenario conditions is also provided.

5.4.2 Frame Sequence Matching

We consider matches between consecutive scenes which are analyzed towards the application of the detectors/descriptors set in vSLAM. This provides insights of the data association process in the location of features from different viewpoints [204, 205].

We use a similar approach to the works reviewed in Section 5.1.3, but since the datasets extract features for real underwater surveys, ground truth of the keypoints was not gathered. Therefore, we apply an inliers ratio criteria based on the inliers obtained after the homography, the number of features detected and the number of matches.

$$Inliers\_ratio\_features = \frac{\#Inliers}{\#Featuresfound} \tag{5.4}$$

$$Inliers\_ratio\_matches = \frac{\#Inliers}{\#Matches} \tag{5.5}$$

5.4.3 Datasets

Data Acquisition

The robot used for the data acquisition is the BlueROV2. All electronics are safely installed in two watertight enclosures rated up to 100 m depth. The top ring holds the logical components, the bottom one the battery. It is equipped with six thrusters set up in a configuration that allows it to move freely within four degrees of freedom (roll and pitch are not controllable) [82].

For communication purposes, the robot has a 100 m tether. This allows easy Local Area Network (LAN) communication even when the robot is in deep water. It also has two Light-Emitting Diode (LED) lights that can be dimmed during use. It has a Raspberry Pi 3 (RPi 3) which takes care of transmitting data from the Remotely Operated Vehicle (ROV) to a connected computer and a RPi 3 camera with wide angle lens which looks straight ahead and can be tilted by a servo by about 45° up and down. Other sensors include a pressure, depth and temperature sensor and an Inertial Measurement Unit (IMU) including gyroscope, accelerometer, and magnetometer. A Pixhawk Autopilot (Px4) controller collects all low-level sensor data and a GPS. Camera images and an extra Xsens IMU are managed by the RPi 3.

This serial port transmits sensor data, robot status and commands using the MAVlink protocol. This is read by the RPi 3 and then sent to a connected client via User Datagram Protocol (UDP) in the original communication setup. This was changed to make sure that data was always sent synchronized and to have a configurable and transparent way of sending, receiving and recording. The protocol to the client was changed to Transmission Control Protocol (TCP) to sync the sensors' reading and the camera reliably. The Xsens accelerometer data is written to a separate file on the RPi 3 and then copied via Secure Copy Protocol (SCP) at the very end of the recording. The modified data flow is shown in Fig. 5.3.

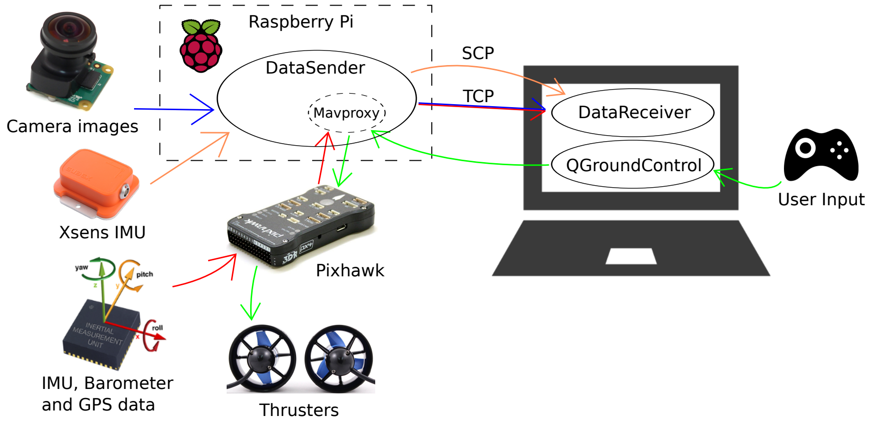


Figure 5.3: Sketch of the modified data flow. Blue arrows represents camera images, red arrows are IMU/barometer data, green arrows symbolize user input and orange arrows are Xsens accelerometer data.

For ease of use the ROV provides three different navigation modes:

Manual Mode: The standard mode in which no stabilization is performed.

Stabilize Mode: In this mode the ROV stabilizes roll and holds its heading (yaw), as long as the user is not trying to turn. Depth control has to be done by the user.

Depth Hold Mode: In this mode the robot behaves as in stabilize mode but also keeps it current depth constant. The user can still control to go up and down but otherwise, the ROV stays at the same depth.

Selected Datasets

We collected different datasets for a variety of underwater scenarios in rivers, beaches, ports and open sea in the surroundings of Perth, Australia[1].

We used the BlueROV2 robot to acquire 1024 x 768 pixels images which are collected on an average of 12 frames per second. Images include part of the structure of the ROV (lights). Eight datasets are selected for the present chapter.

In Table 5.3, the selected datasets are described based on the underwater alterations explained in Section 5.3. The datasets covered sandy and rocky backgrounds with the presence of algae, far algae means that the algae is viewed as patches or are not moving, close algae means that algae is observed closely and movement is captured. Some datasets recorded isolated objects such as poles, rocks, part of a wreck and debris. The symbols >>, >, <, << are used to indicate the quantity. The rotating over an object cell point out the frames involved in the navigation of the ROV around an object (frames in thousands).

Table 5.3: Datasets characteristics

	Dataset_1	**Dataset_2**	**Dataset_3**	**Dataset_4**	**Dataset_5**	**Dataset_6**	**Dataset_7**	**Dataset_8**
Seafloor	sandy, algae (far), algae (close)	sandy, algae (far)	rocky, algae (far)	sandy, <algae (far)	sandy, algae (far)	sandy	sandy	sandy
Objects		poles		rocks	wreck	<debris	>>small rocks	<partial poles
Light	non-uniform	>>uniform	caustic pattern	night, backscatter	<<non-uniform	caustic pattern	>>uniform	>>uniform
Tint	greenish	greenish	natural	natural	greenish	natural	natural	greenish
Turbidity	low	low	low	low	low	low	low	low
# Frames	11729	5830	1929	8308	9155	2514	2522	2388
Notes		horizontal and vertical poles	little algae on rocks			robot shadow		wavy pattern on sand

5.4.4 Experimental Setup

We used a desktop computer with an Intel® Core™ i7-7500U CPU @ 2.70GHz × 4 CPU and 16GB of RAM with Ubuntu 16.04 for the evaluation. The OpenCV [206] implementation of SIFT & SURF (non-free module xfeatures2d), AKAZE, ORB and BRISK are used. As well as the Nearest Neighbour (NN) algorithm for detecting matches between keypoints sets and Homography based on Random Sample Consensus (RANSAC) to reject outliers. The evaluation setup is based on the work found in [207] which integrates the OpenCV implementations in a friendly user GUI.

[1] http://robotics.ee.uwa.edu.au/auv/ftp/Underwater_datasets.zip

The modified program follows the block diagram presented in 5.4 to perform our evaluation. The datsets are masked to exclude the lamps from the ROV which are easily recognizable by the detectors and appears in every frame causing inconsistencies in the matching process. The data was logged into Comma Separated Values (CSV) files keeping the record of the number of features found, matches and processing time.

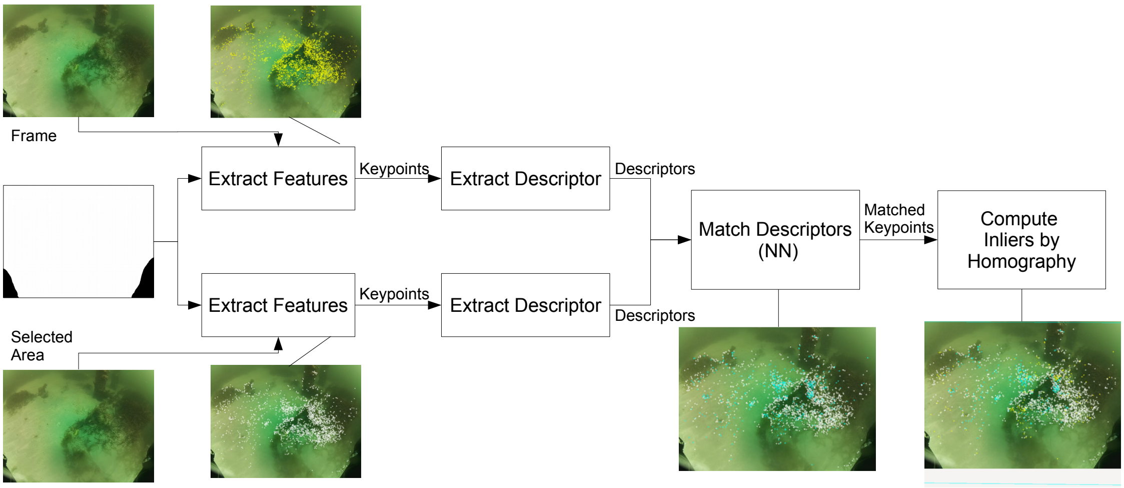


Figure 5.4: Block diagram of data extraction for evaluation

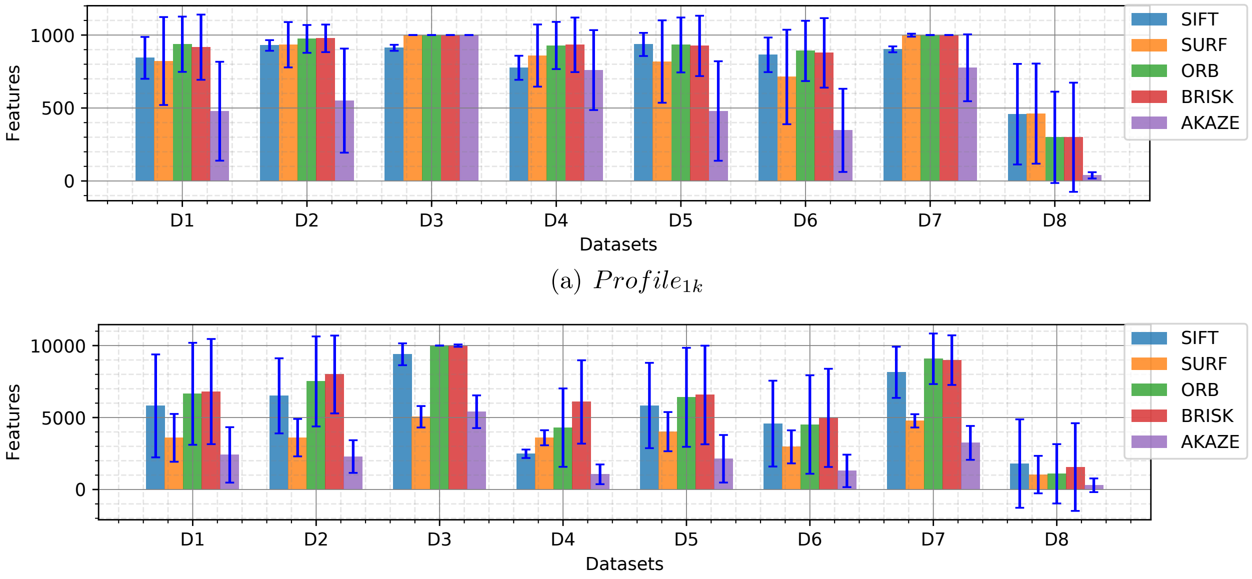
5.5 Results and Discussion

5.5.1 Detectable Features in Underwater Images

In Fig. 5.5 an overview of the features extracted per dataset is shown. The bar graphs show average values and the standard deviation to quantify the dispersion of the values obtained. At this point, the number of features only indicate that the images present detectable salients, but not if they are going to be recognizable in the following frames. This information is still useful to identify which underwater elements are detectable.

The overview shows an overall homogenous performance detecting around 500 features in $Profile_{1k}$ and 5000 features in $Profile_{10k}$ for all the detectors. Dataset_1 shows a high dispersion of the data for most of the detectors. Datasets 4, 6 and 8 present a low average compared to the rest. It is worth mentioning that the detailed graphs for both graphs have similar behavior, the only difference is the number which is proportional to the maximum number of features per profile. Therefore, in most cases, we analyze the $Profile_{1k}$ detail where the fluctuations, when finding a low number of features, are more evident than in $Profile_{10k}$.

We have selected two datasets to show the performance of the feature detectors in the underwater scenario. In Fig 5.6 (a) can be seen the performance of the detectors applied to Dataset_1. Algae offer a good contrast on the sand exposing detectable features as seen in Fig 5.6 (b-f), it can be seen how ORB, BRISK, SIFT and AKAZE features surround the algae while SIFT features are more spare along the entire image. The figures also show that the detectors cannot find many features in plain sandy areas. During the frames ∼3000 – ∼4200 the ROV gets far from the seafloor, and the algae are seen as blurry patches, in this case, none of the detectors were able to extract much features (Fig 5.6 (g,h)).



(a) $Profile_{1k}$

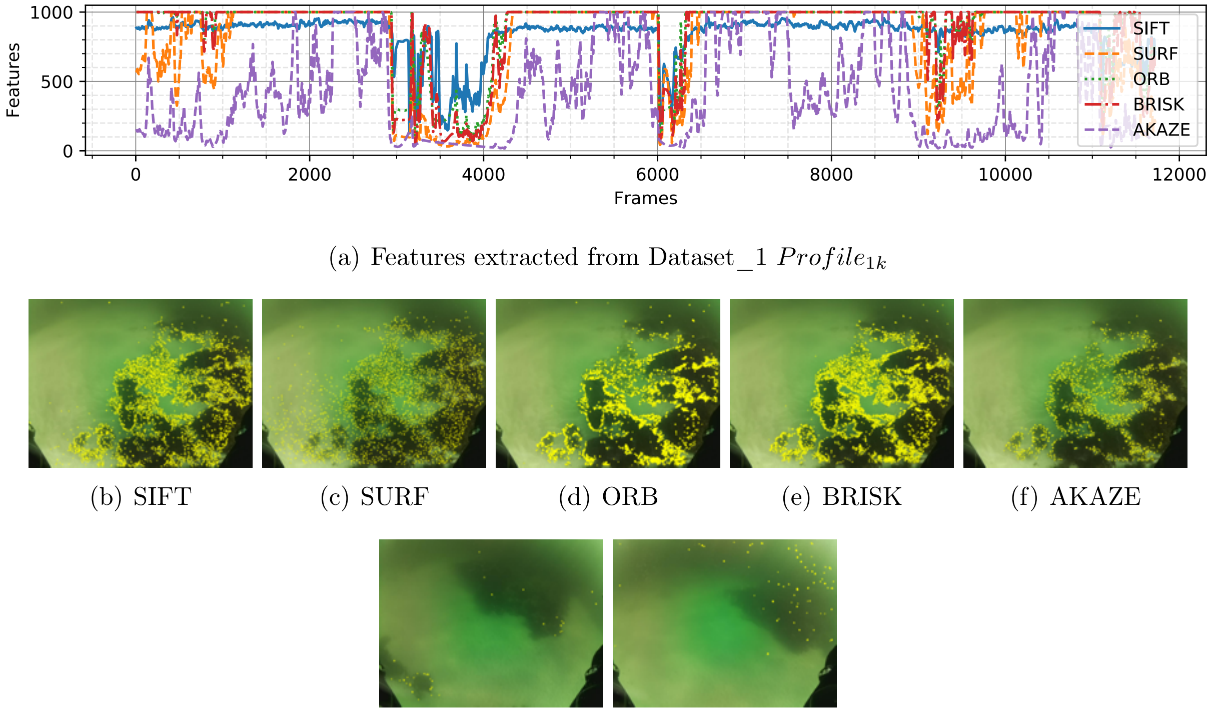
(b) $Profile_{10k}$

Figure 5.5: Features extracted per dataset

Dataset_8 is mostly sandy with some frames capturing partial poles as objects. The illumination is uniform and has a greenish tint (Table 5.3). As observed in Fig. 5.6, plain sandy areas are hard environment to extract features from. Fig. 5.7 shows the detail for Dataset_8. When the robot is close to the seafloor (20cm approximately) the detectors start extracting features from the wavy pattern of the sand.

5.5.2 Frame Sequence Matching

It is important to quantify the number of features that can be re-observed (matched) in the following frames for the vSLAM scope. In Fig. 5.8 a bar graph of the inliers obtained after applying NN and homography with the consecutive frame is shown. In this test, the descriptors obtained from the keypoints found with the detectors are evaluated. Similarly, in the number

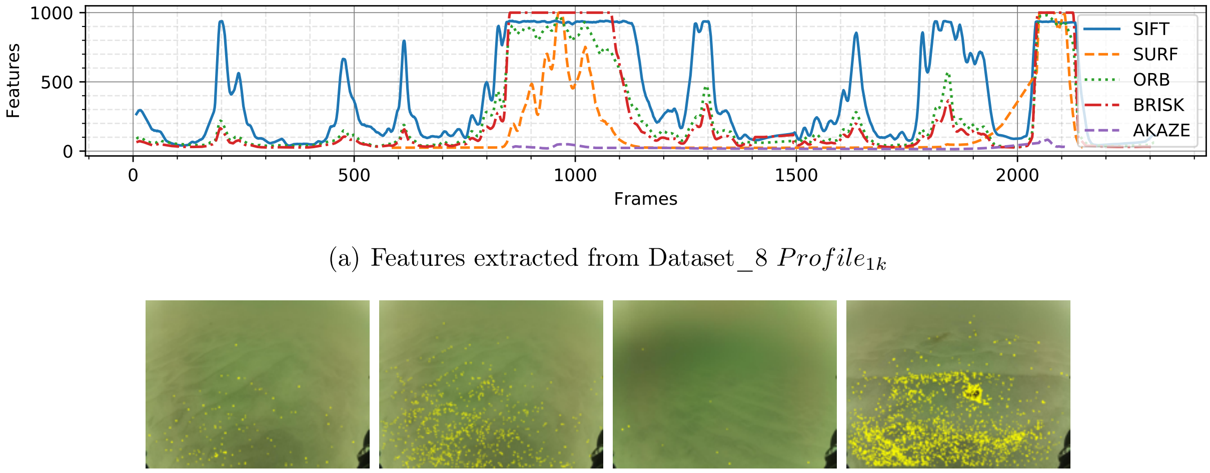


(a) Features extracted from Dataset_1 $Profile_{1k}$

(b) SIFT (c) SURF (d) ORB (e) BRISK (f) AKAZE

(g) ORB f3148 (h) SURF f3550

Figure 5.6: Detail of features extracted from Dataset_1. b-f, frame 710 with features extracted, g and h show low features found

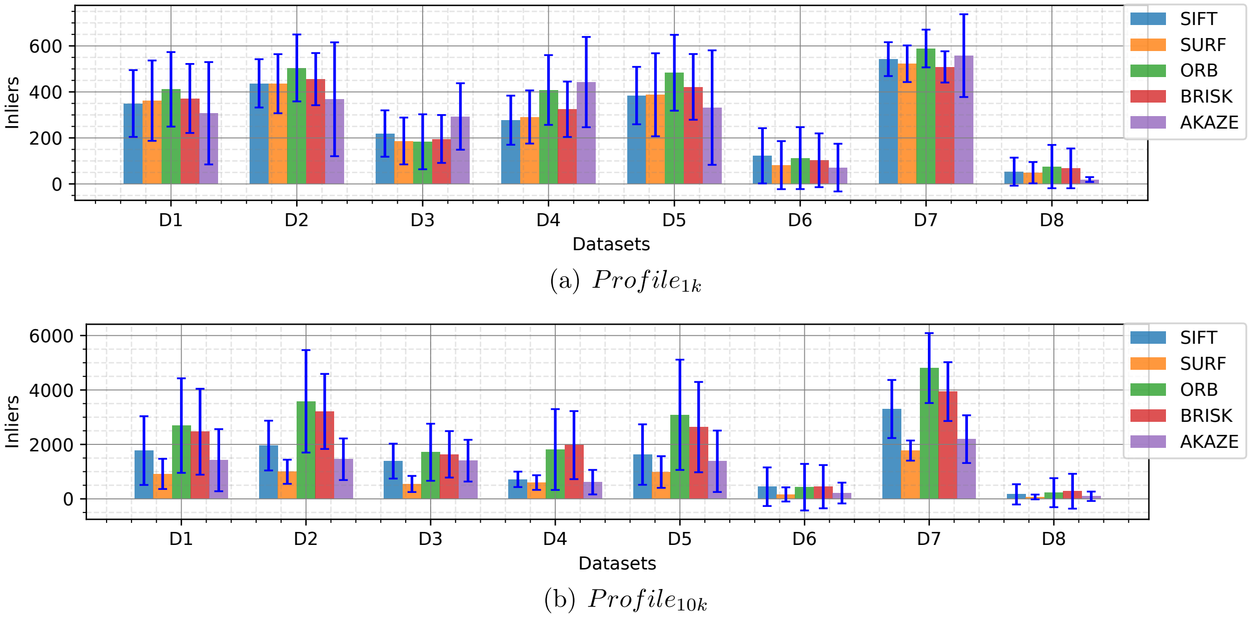


(a) Features extracted from Dataset_8 $Profile_{1k}$

(b) ORB f198 (c) SIFT f198 (d) SURF f350 (e) ORB f1017

Figure 5.7: Detail of features extracted from Dataset_8. b-e, extractors application in different frames

of features found, $Profile_{1k}$ and $Profile_{10k}$ show similar behavior for the different detectors, and the results are homogeneous among them. Datasets 3, 4, 6 and 8 present the lower average number of inliers. In the case of $Profile_{10k}$ SIFT, ORB and BRISK features slightly stick out compared to the others, especially in Datasets 2 and 7. AKAZE, which showed a lower number of features extracted in Fig. 5.5, shows around the same amount of inliers than the others.



(a) $Profile_{1k}$

(b) $Profile_{10k}$

Figure 5.8: Inliers, obtained after NN matches and homography, per dataset

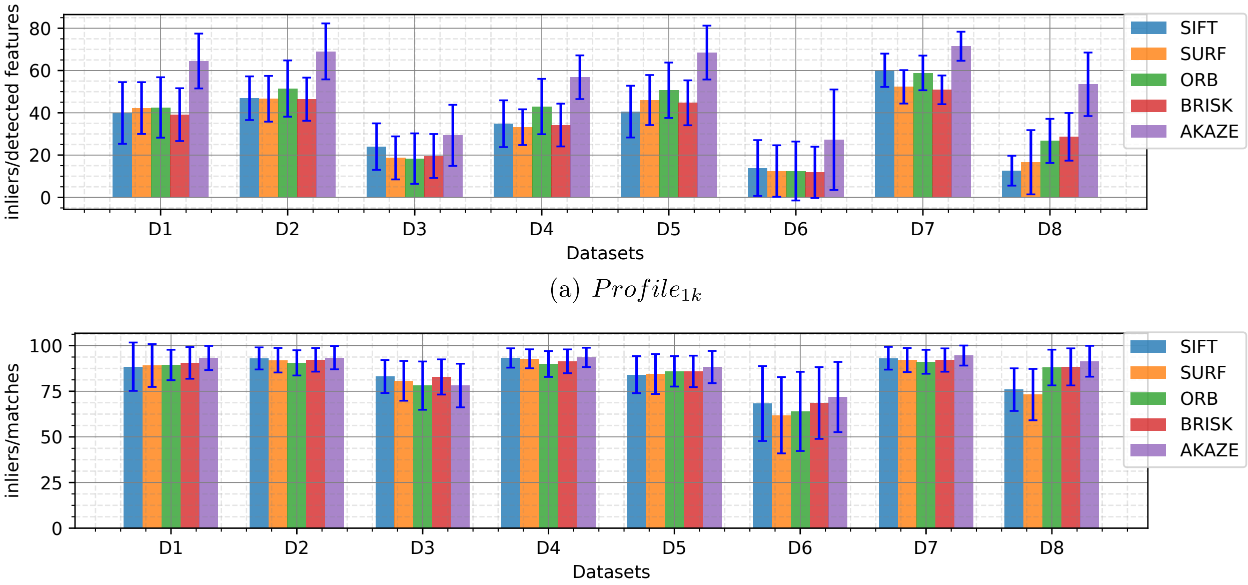
Fig. 5.9 shows the ratios presented in Eq. 5.4 and Eq. 5.5, in percentage, for $Profile_{1k}$. In Fig. 5.9(a) it can be seen that around 40% of the features found by the detectors are matched correctly in the consecutive frame. AKAZE outstrips the other extractors/descriptors in the performance, demonstrating that its extractor is more finicky than the others.

In Fig. 5.9(b) can be observed that more than 75% of the features matched become inliers after homography indicating a good performance overall for the descriptors evaluated.

5.5.3 Image Enhancement

We applied two image enhancement algorithms for underwater images to Datasets 3, 4, 6 and 8 which showed the lowest number of features or inliers found. In Fig. 5.10 the enhancement by fusion filter [203] is represented by an ‘F’, and the backscatter removal filter [195], by a ‘B’. The results without any enhancement are shown in grey for easy comparison.

The number of features extracted increases for Datasets 4, 6 and 8 (Fig. 5.10(a)). Dataset_3, which is affected by light caustic patterns on a rocky background, does not show any improvement



(a) $Profile_{1k}$

(b) $Profile_{10k}$

Figure 5.9: Inliers ratios per dataset

by any of the two algorithms. The image enhancement algorithm by fusion shows a better result exposing detectable features for the detectors.

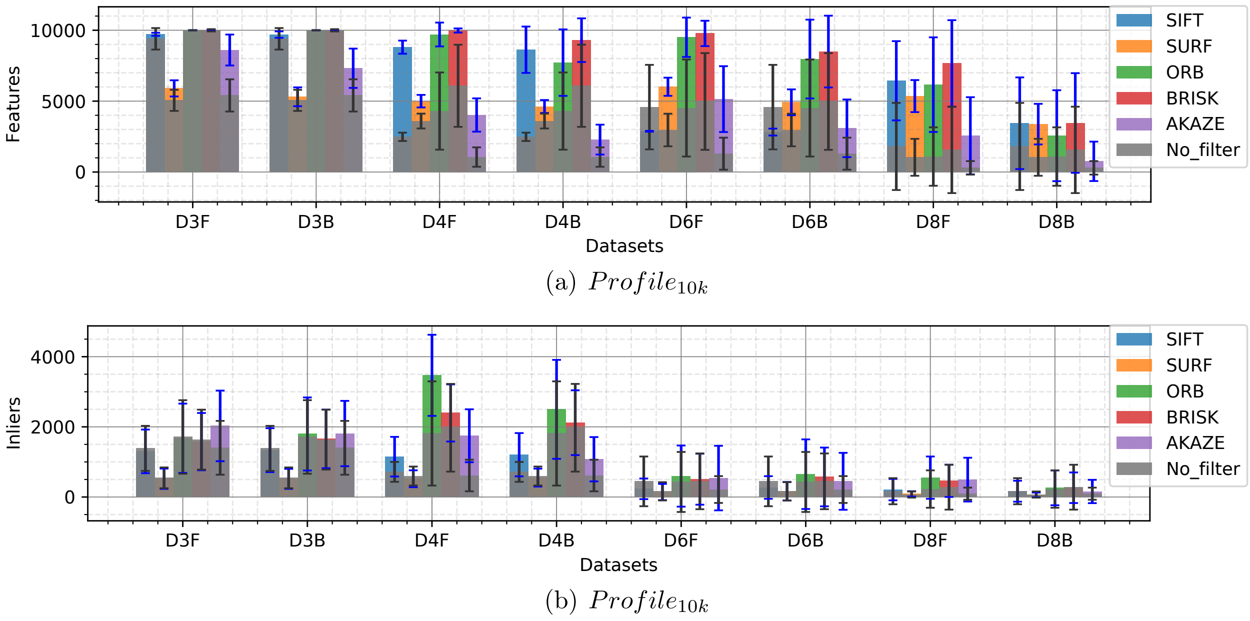
It can be seen in Fig. 5.10(b) that, in the case of SURF descriptors the number of features found presented and increase although, this increase is not observed at the time of matching those features in the consecutive frames. AKAZE benefits the most from the enhancement algorithms showing an improvement for all datasets. ORB, SIFT and BRISK are also helped by the algorithms in the order presented.

Dataset_4, which was taken at night with artificial illumination on a sandy background with few algae and rocks, gets the most significant improvement in the number of inliers. The filter by fusion gets better results than the backscatter filter.

Datasets 6 and 8 also increase their number of inliers, especially with the filtering by fusion. These two scenarios present a sandy background with few objects on the seafloor. Both present illumination problems, Dataset_6 presents a caustic pattern and Dataset_8 a non-uniform illumination.

5.5.4 Processing Time

The processing time is measured for the detection and describing, NN matching and homography for the two profiles. In Fig. 5.11, the processing time for Dataset_2 is presented which also includes the pre-processing time for the enhancement algorithm.



(a) $Profile_{10k}$

(b) $Profile_{10k}$

Figure 5.10: Results for pre-processed Datasets 3, 4, 6 and 8

ORB is the fastest set detector/descriptor with an average processing time of 43ms and 97ms for $Profile_{1k}$ and $Profile_{10k}$ respectively. SIFT and BRISK are the slowest with times around 150ms and above 300ms for $Profile_{1k}$ and $Profile_{10k}$. BRISK presents the highest dispersion having variations correlated with the number of features found, similar to SIFT; the rest show a continuous time for processing.

The enhancement algorithms applied are highly time-consuming showing values above 1 and 2 seconds for the algorithms filtering by fusion and backscatter removal respectively.

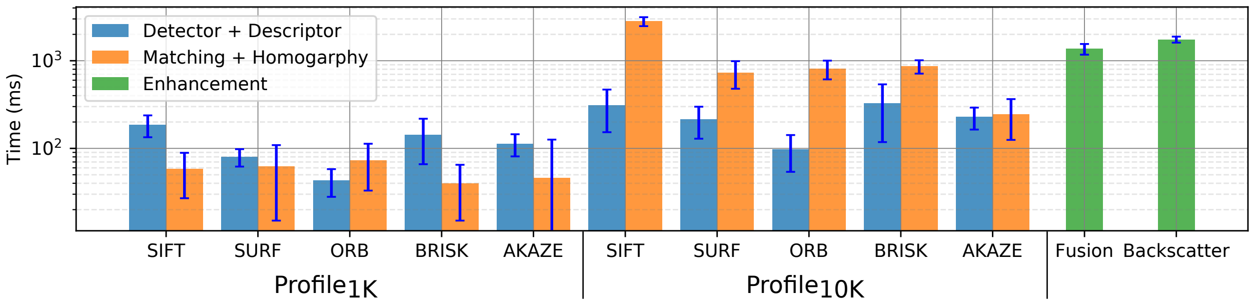


Figure 5.11: Processing time based on Dataset_2

5.6 Chapter Summary

The experimental results provide a detailed analysis of SIFT, SURF, ORB, BRISK, and AKAZE detectors/descriptors for underwater environments towards the application of vSLAM.

The detectors selected in this survey showed a satisfactory performance on images containing color distortion, low non-uniform illumination and low turbidity. Sandy environments with algae patches, algae recorded from close and far, small particles such as debris and rocks, and objects such as poles and rocks present detectable features for the extractors.

Different datasets were categorized according to the characteristics of the seafloor, types of objects, lighting, tint, and turbidity. The influence of these effects on the images is observed in the number of features extracted and later matched in subsequent frames. The results showed decreased of features and matches due to turbidity, blurriness, in Fig. 5.6 (a)(g)(f); monotony, sand patches with and without texture, in Fig. 5.7; and lighting, caustic patterns, shown in the overall number of features (Fig. 5.5) and in the number of matches (Fig. 5.8).

The number of inliers when matching keypoints from consecutive frames was homogeneous among the detectors, in $Profile_{10k}$ ORB and BRISK stick out. AKAZE achieved a better ratio of inliers/detected_features.

The two enhancement algorithm applied in this survey showed an improvement in the performance of the detectors/descriptors. The filter by fusion [203] showed the higher improvement especially in night scenarios with artificial light, caustic pattern and significant non-uniform illumination.

The survey provides abundant information and detailed insights valuable for making decisions in applications towards vSLAM. The ORB detector/descriptor stood out in detection and matching performance, shaping up as a good selection for implementing vSLAM, with the lowest computing time.

Chapter 6

Experimental Evaluation of Monocular ORB-SLAM2 in Underwater Environments

This chapterpresents an experimental evaluation of monocular ORB-SLAM2 applied to underwater scenarios. For this purpose, we collected more than 40 datasets in different areas and with varying weather conditions. Underwater images present challenges for image processing such as monotony, turbidity, dynamics, and lighting variations which affect the performance of the algorithm. Our results show a low impact of turbidity and dynamics; mid-impact of monotony, especially in low texture; and high impact of lighting and flickering, observable on sunny days, which tend to disappear below the 3 meters of depth. To improve performance under these circumstances we provide possible algorithm enhancements.

6.1 Introduction

Over the years, visual Simultaneous Localization and Mapping (SLAM) has been applied in a multitude of robotics scenarios, including in the air using Unmanned Aerial Vehicles (UAVs) [208]; on roads with autonomous cars [209]; inside buildings [210] or even on Mars [211]. Another scenario that has received a lot of interest is the underwater environment, but due to the difficult conditions of this setting it is still considered “an unsolved problem in robotics” [212]. Commonly used platforms for SLAM underwater research are Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs). These robots often come with a single camera and few other sensors for navigation and data collection.

Over the last years, many different visual SLAM approaches such as Dense Tracking and Mapping (DTAM) [213], RatSLAM [214], Large Scale Direct (LSD) SLAM [215] and ORB-SLAM2 [216] have been developed [161]. Unfortunately, these algorithms are mostly evaluated on land without considering applications in underwater environments. The underwater scenario presents different challenges such as monotonous areas, turbidity, light scattering, changes in colors and highly dynamic movement of the robot when capturing the images. Li et al. [175] were able to show that ORB-SLAM2 is capable of performing well in different scenarios, including underwater. Unfortunately, to the best of our knowledge, no research has been done on an exhaustive performance analysis in terms of the application of visual SLAM in underwater environments and its challenges.

To evaluate the impact these characteristics have on the results of a state-of-the-art algorithm, we conducted a detailed evaluation of ORB-SLAM2 in underwater environments, considering different scenarios; lighting-conditions; structured areas such as pools, decks, shipwrecks and marinas; and unstructured areas such as open sea and rivers. Additionally, weather conditions varying from sunny to cloudy days are also included in the datasets. The results presented can be considered as a baseline for evaluating future improvements to ORB-SLAM2 or other visual SLAM methods.

This chapter is organized as follows. In Section 6.1, an introduction to the chapter is given. In Section 6.2, other works which investigate underwater visual SLAM approaches are outlined. Section 6.3 gives an overview over the ORB-SLAM2 algorithm and the challenges it has to face underwater. In Section 6.4 we present our experimental setup, the performed experiments and how we evaluate them. The insights gained from these experiments are given in Section 6.5. Finally, the conclusion is presented in Section 6.6.

6.2 Related Work

A monocular camera setup in an underwater environment has only been tested in a few research papers so far. Li et al. [175] compared different open-source SLAM and visual odometry algorithms on eight different datasets. These datasets included data recorded in both underwater and land environments using different means of recording. Two of these datasets were recorded with an AUV, one in a coral reef and one inside a wreck. The other two sets were recorded using handheld cameras. In one, a camera was mounted on a drifter and moved by the waves alone, in the other, the camera was moved manually. The best results on the underwater datasets were achieved by Parallel Tracking and Mapping (PTAM) [217] and Oriented FAST and Rotated BRIEF (ORB) SLAM. Li et al. also state that “ORB-SLAM2 is the package that provides the best results in terms of accuracy" [175].

Concha et al. [218] proposed a semi-dense approach similar to LSD SLAM which incorporates the ability to differentiate between image areas with poor visibility and those with good visual information. They did not use any form of optimization on their pose-graph and could therefore only map small areas. No information regarding the accuracy of the algorithm is provided.

In [159] Negre et al. explore a new way of detecting loop closures. By forming and recognizing feature clusters they were able to show superior results in comparison to ORB-SLAM2's loop closing approach. Since their implementation relies on a stereo camera setup it is not directly applicable to monocular frameworks.

Chaves et al. [219] used an active SLAM approach which tries to increase the localization accuracy by actively searching for trajectories which maximize loop closures. They used a saliency prediction which helps create paths the robot can use for finding good loop closing opportunities.

Yet another approach has been put forward by Silveira et al. In [220] they explain their DolphinSLAM approach which is derived from RatSLAM. Based on insights from [221] their front-end uses Speeded-Up Robust Features (SURF) for feature detection. They are able to show that the algorithm is capable of localizing an AUV in different environments. This is also not a purely camera driven SLAM variant since they incorporate a Doppler Velocity Log (DVL) and sonar into their framework.

6.3 ORB-SLAM2 and the Underwater Scenario

6.3.1 Algorithm Description

ORB-SLAM2 derives its name from the Oriented Fast and Rotated Brief (ORB) feature descriptor which was developed by Rublee et al. as "an efficient alternative to Scale Invariant Feature Transform (SIFT) and SURF" [181]. According to Mur-Artal et al. these features provide "good invariance to changes in viewpoint and illumination" [216] and are cheap to compute. As is obvious from the use of ORB features, ORB-SLAM2 uses a feature-based front-end. The back-end works on a keyframe-based graph-optimization procedure.

Mur-Artal et al. utilize three parallel threads: tracking, local mapping and loop closure as shown in Fig. 6.1 described in [216]. The tracking thread is responsible for tracking the movement of the camera. In a first step, this thread extracts the ORB features. Then, these features are matched with those of the previous frame. If the matching is successful, a constant velocity model is used to estimate the new camera pose. If it fails, a relocalization based on a place recognition database is initialized.

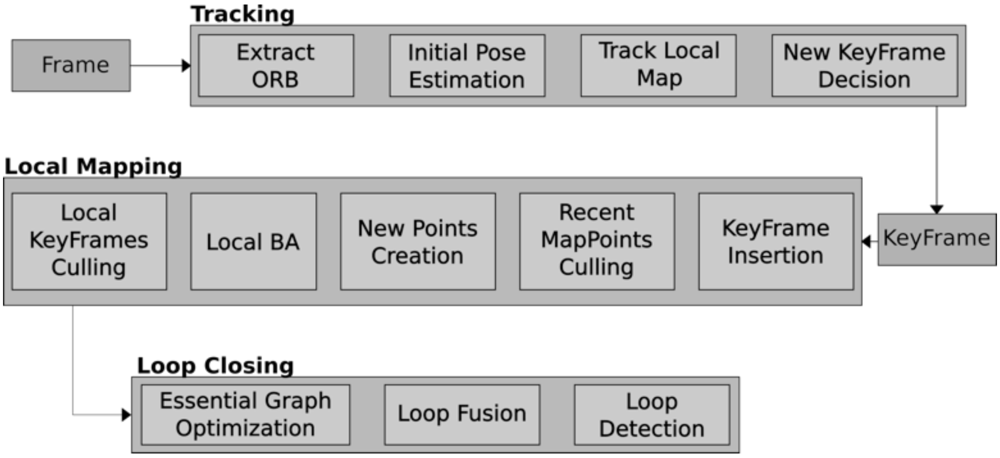


Figure 6.1: ORB-SLAM2 Overview

Once the initial estimate has been retrieved, the local map is searched for correspondences between image features and map points. The local map consists of keyframes K_1 which share map points with the current frame and those which neighbor K_1 keyframes in a space graph called *covisibility graph*. Doing this the estimated pose can be optimized with minimal computational cost [216]. Based on the amount of new features in the image the algorithm then decides whether the current frame is used as a new keyframe. The local mapping thread keeps the local map up to date by inserting new keyframes into the *covisibility graph*. It also checks whether new map points are being tracked in following keyframes. If not, these are removed. By applying a local Bundle Adjustment (BA), the current keyframe, all those keyframes connected to it and all map points seen by those connected keyframes, are optimized. In order to keep the computation costs low redundant keyframes are gradually removed from the map.

Once local mapping is done with a keyframe, it is passed on to the loop closing thread. Using this keyframe a search for loop closing candidates is started in the place recognition database. These candidates are then evaluated and, if the loop is accepted, it is used to optimize the complete graph.

Bundle Adjustment

Bundle Adjustment is an optimization algorithm based on least squares optimization of error functions (linear and nonlinear) that can be represented by a graph [222].

It is widely used in visual SLAM for minimizing the re-projection error in two views where the two camera poses and the 3D points corresponding the 2D gathered from both cameras have to be optimized to fit through measurement function.

In ORB-SLAM2 the open source library g2o for BA is used in different modules such as in local graph for pose bundle adjustment, co-visibility graph for local bundle adjustment, and essential graph for global bundle adjustment after loop closure detection.

6.3.2 ORB-SLAM2 Validation

It is difficult to measure complete ground truth even in indoor environments for evaluating visual SLAM in terms of the robot pose and landmark positions [169]. Ground truth robot pose instrumentation include 3D laser scanners and a high-precision GPS/IMU as in the case of the *KITTI* dataset [223]. For underwater environments acoustic positioning systems have to be added such as Ultra-Short BaseLine (USBL) [224] which requires expensive external instrumentation.

Localization for ORB-SLAM2 has been evaluated by their creators in [225] using ground truth datasets, including *KITTI*. Monocular, stereo and RGB-D datasets are evaluated with ground truth and achieve very robust localization results especially for stereo and RGB-D. ORB-SLAM2 accomplish zero-drift localization for well mapped areas. Monocular cameras do not capture depth, the scale of the map and the trajectories are unknown, therefore, it requires a posterior adjusting of the scale. In the case of Monocular ORB-SLAM2 scale drift might occur, principally at the turns.

In [174], the authors evaluate ORB-SLAM2 among other visual SLAM algorithms. After scaling the trajectory manually to obtain the smallest error, it obtains a root-mean-square error (RMSE) of 0.05 m with a standard deviation (σ) of 0.02 m for the best case. Conversely, it obtains a Root Mean Square Error (RMSE) of 1.1 m with a σ of 0.02 m for the worst case.

6.3.3 Underwater Challenges for Visual SLAM Approaches

Underwater images captured by a standard monocular camera bear many challenges (Fig. 6.2) which are not present on land. Related works with underwater images noted bad illumination, sand patches, light scattering and turbidity [187, 188].For visual SLAM (vSLAM), challenges include:

- **Monotony:** In many locations the sea floor consists mainly of sand and does not offer many recognizable structures which could be used for extracting landmarks. This may cause some SLAM algorithms to fail.
- **Turbidity:** Seawater generally has a lot of different particles in it that can cause trouble when using cameras. Feature-based approaches might falsely detect particles as features which can have strong effects on the algorithm's accuracy.

- **Dynamics:** Water tends to be highly dynamic so even if the robot is not actively trying to move, it is still subject to currents and water movement. Many SLAM algorithms rely on a kinematic model to predict the current pose which needs to take the water dynamics into account.
- **Loss of colors:** Due to the effect that water absorbs different wavelengths of light at different depths the visual appearance of the environment quickly becomes monotonous. Color gradients can be a valuable information source which might not be available underwater.
- **Lighting:** Not only does water absorb light, any movement of the water surface will also cause the light to scatter and flicker. This leads to dynamic shapes on the ground which make it difficult for an algorithm to detect static landmarks.

6.4 Evaluation

6.4.1 Dataset Contents

The main goal while recording datasets was to cover as many different scenarios as possible so that the ORB-SLAM algorithm could be tested with varying conditions. For that reason the recorded scenarios include:

- Man-made structures like pool, jetties, pipes and boats.
- Natural environments like reefs, river beds or sea floor.
- Night and day settings.
- Varying depths.
- Sunshine and cloudy weather.

The ROV was driven in different patterns like rectangles, circles or an eight which makes it easier to determine if ORB-SLAM is estimating the correct trajectory. In total, 46 datasets were recorded in nine different locations[1].

A mask was implemented to define a region of interest excluding the parts of the images where the robot sees its own. Without it, ORB-SLAM2 extracts and tracks features in those areas and lead to frequent tracking failure.

[1] `http://robotics.ee.uwa.edu.au/auv/ftp/Underwater_datasets.zip`

6.4.2 Evaluation Criteria

The evaluation of the algorithm performance is divided in a quantitative and qualitative analysis. The first one is a statistical overview of the tracked frames and, the qualitative, represents the performance of the trajectory and map generated according to how the robot was driven.

Due to heavy use of Random Sample Consensus (RANSAC) [226] and multi-threading, ORB-SLAM2 is a highly non-deterministic algorithm. To account for this, every experiment was run and evaluated ten times (Appendix B). The percentage of the tracked frames (*TF%*) and the number of loss of tracking per thousand images over all the ten tests (*LT/1000*) will be given.

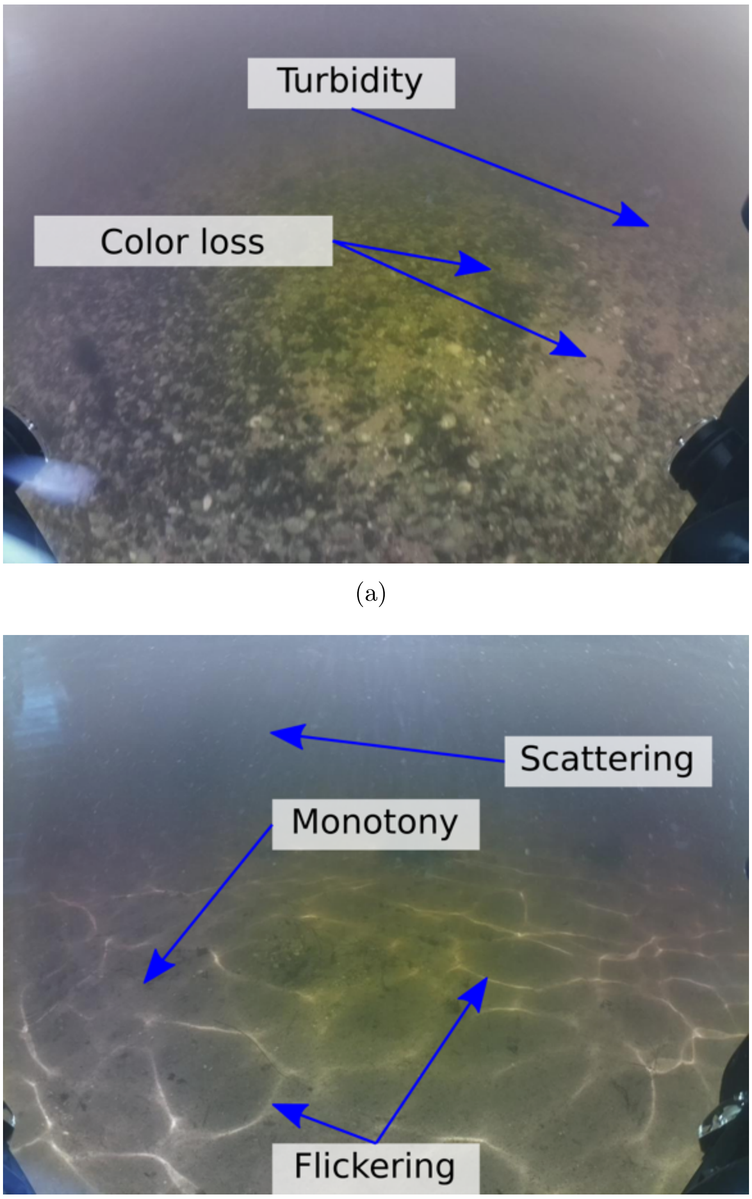
For each bar plot the most representable of all ten tests is picked. The bar represents ORB-SLAM2's states: initializing (light yellow), tracking (green) and relocalizing (red) plotted over the frames of an experiment. Additionally, found loops are marked by a blue bar and an *'L'*. In some cases manual resets were performed (marked by an *'R'*), when the algorithm enters to a forever relocalization loop (Fig. 6.3).

As discussed in Section 6.3.2, proving ground truth is difficult and expensive; therefore, to provide references to evaluate the performance, physical references and driving in known shapes were used. These include driving on the edges of a pool with known dimensions as well as going in circular or rectangular patterns. But, even if there was a physical reference, waves, surge and inaccurate robot controls made it difficult to follow them precisely. In light of this issue the evaluation performed here cannot be based on calculating an error between ground truth and estimated trajectory.

Instead, a Qualitative Validation of the Trajectory (QTV) is used for evaluating ORB-SLAM2. It splits results into three categories as follows:

- **Good**: Both the estimated trajectory and the map closely represents the robot's movement and the actual environment without any noticeable deviation.
- **Acceptable**: Estimated trajectory and map still very much represent the robot's movement and environment but they might have minor notable inaccuracies.
- **Poor**: This will be chosen for major unresolved inaccuracies in either map and/or trajectory and in cases of falsely detected loops or false relocalization.

The following sections will describe and evaluate the results of the experiments by location. For reasons of space not every single dataset is described, but rather those which provide the most information.



(a)

(b)

Figure 6.2: Examples of challenges for Visual SLAM

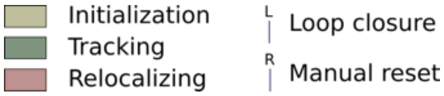


Figure 6.3: Bar plot elements

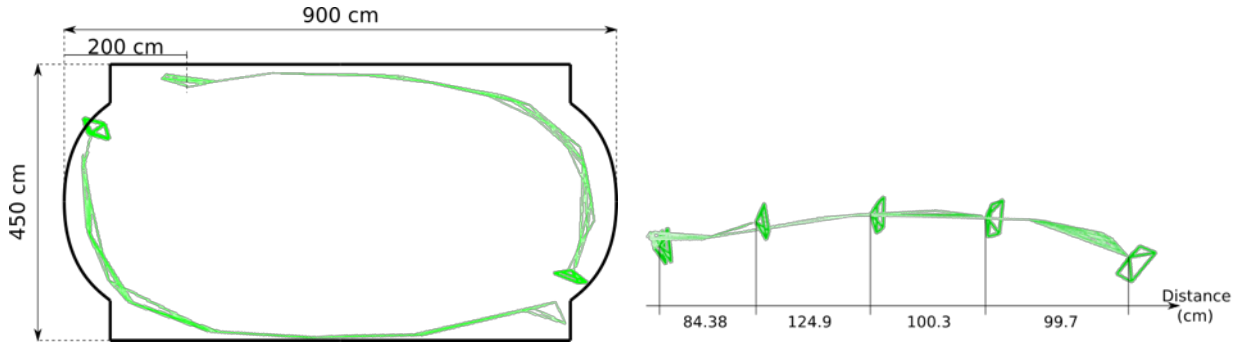
6.4.3 Experiments

In Section 6.2 the need of a posterior scaling for evaluating ORB-SLAM was presented. The first experiment was performed to give an estimate of ORB-SLAM's accuracy in underwater. For this purpose the robot was driven in a pool with known dimensions. Since monocular SLAM is not able to measure any scale, the resulting trajectory was manually scaled to fit the pool's shape before calculating the error. The experiments from 4.2.2 and onwards use the evaluation criteria described in Section 6.4.2.

Pool—Validation

This experiment was carried out in a recreational pool with a plain bottom (no tiles or lines draw) in which 50 landmarks were placed. Fig. 6.4 (a) shows the path of two consecutive tests around the pool. The proportions of the length and curvature of the trajectory matches the pool once scaled.

An additional test was performed taking measurements every five seconds while driving the robot along one of the edges. For evaluating this test, the trajectory was scaled manually resulting in an RMSE of 9.8 cm and a σ of 6.6 cm. It is worth mentioning that measurements are approximate as the evaluation was performed by marking the pool's edge while the robot was moving and then measuring the distances using a measuring tape.



(a) Estimated robot trajectory in pool

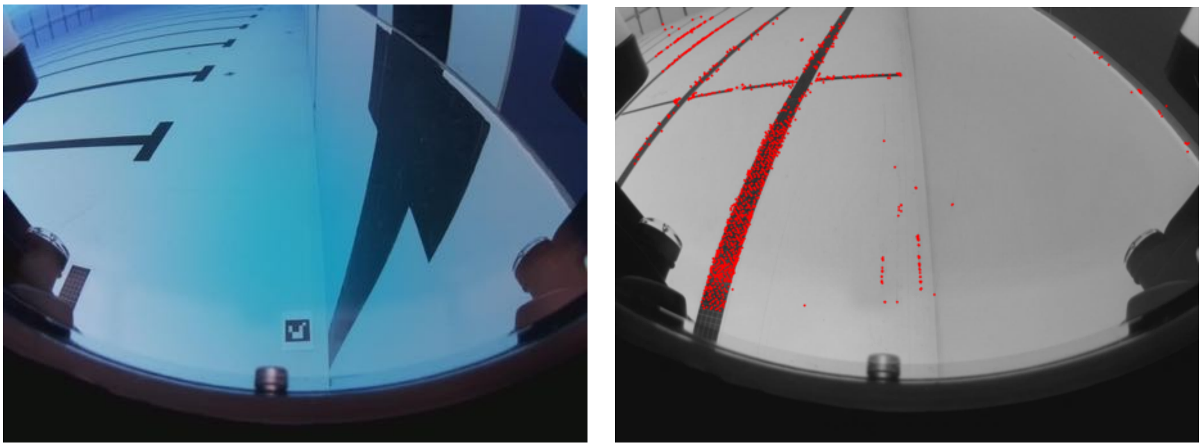
(b) Distances test

Figure 6.4: ORB-SLAM validation test in a recreational pool

Pool - The University of Western Australia (UWA) pool

Description:

The UWA pool has a rectangular shape with dimensions 33.5 m by 25 m. Inside the pool, there are black lines made of tiles on both wall and floor. Since it was suspected that ORB-SLAM2 might have trouble working in this very symmetric and feature scarce environment, one experiment was done with 20 unique printed markers placed on the pool's floor (Fig. 6.5 (a)). In order to give the algorithm enough chances for loop closures, the ROV was driven along the pool's walls in both datasets whilst always staying on the surface.

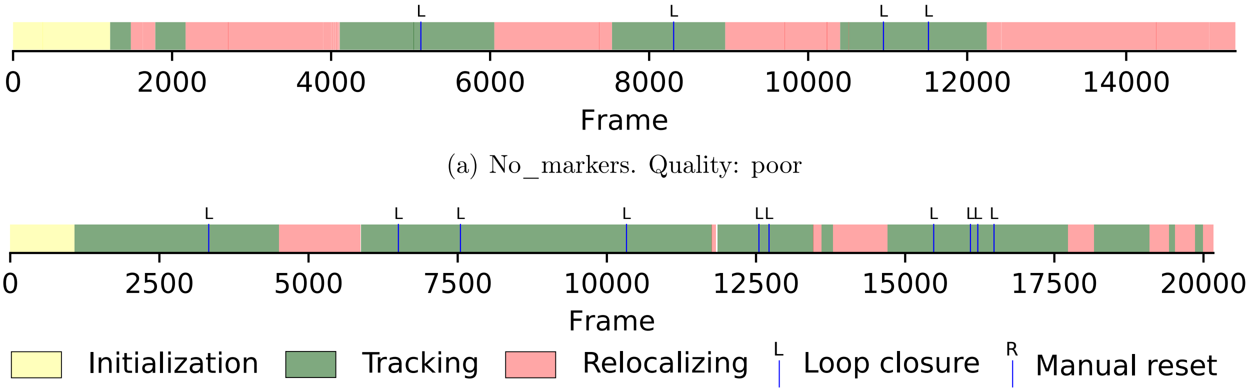


(a)

(b)

Figure 6.5: Pool dataset: (a) sample image. (b) example of poor feature distribution

Evaluation:



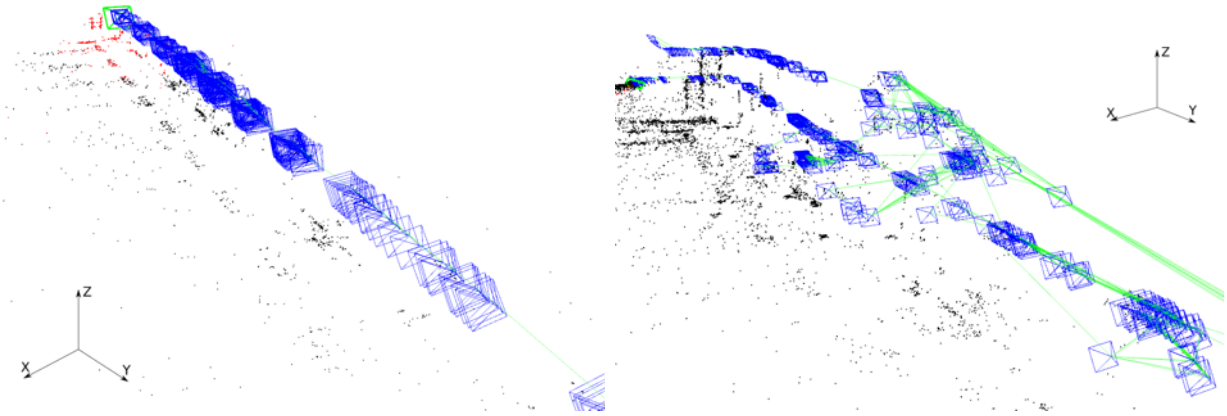
(a) No_markers. Quality: poor

(b) Markers. Quality: poor

Figure 6.6: UWA Pool selected bar graphs of ORB-SLAM2 states

The UWA pool proved to be a very difficult environment for ORB-SLAM2 to work in. Figure 6.6 (a) illustrates several of the problems encountered. The periodical tracking loss and regain shows that while the robot is driving along the wall where it is moving orthogonal to the black lines on the pool floor, as shown in Fig. 6.5 (a), it can mostly maintain tracking. While driving along the other wall where the ROV follows the black line on the floor, as in Fig. 6.5 (b), tracking is lost very quickly. Tracking is most likely lost because of the poor feature distribution shown in Fig. 6.5 (b). Since there are lots of features on the black line which barely differ and do not change much over time ORB SLAM fails to properly associate the seen features between frames. The results for the dataset with markers proved to be just as unusable as those of dataset without markers due to the features distribution was still predominant on the lanes and not much on the markers. Dataset $Markers$ has a $TF\%$ of 75% and dataset $No\_markers$, of 40%; in terms of $LT/1000$, dataset $Markers$ loses tracking around 3 times compared to dataset $No\_markers$ which loses tracking 1.5 times. The markers help to reduce the times tracking is lost ($LT/1000$), but the overall results are still poor.

The resulting trajectories for the two datasets are shown in Fig. 6.7. For the dataset without markers it is obvious that ORB-SLAM2's estimated trajectory is only a single line even though it was tracking on two sides of the pool. What happens is that, due to the symmetry of the pool, ORB-SLAM2 relocalizes within the map created along the first wall even though it is on the exact opposite side of the pool. The problem with the repetitive environment is further demonstrated by the fact that the first loop is found before a full path around the pool was driven. Due to relocalization and loops detected ORB-SLAM2 simply keeps jumping back and forth within the map shown in Fig. 6.7 (a). A similar result is found in Fig. 6.7 (b) since the markers did not help much, although some displacement orthogonal to the lanes was tracked but still, not completely.



(a)

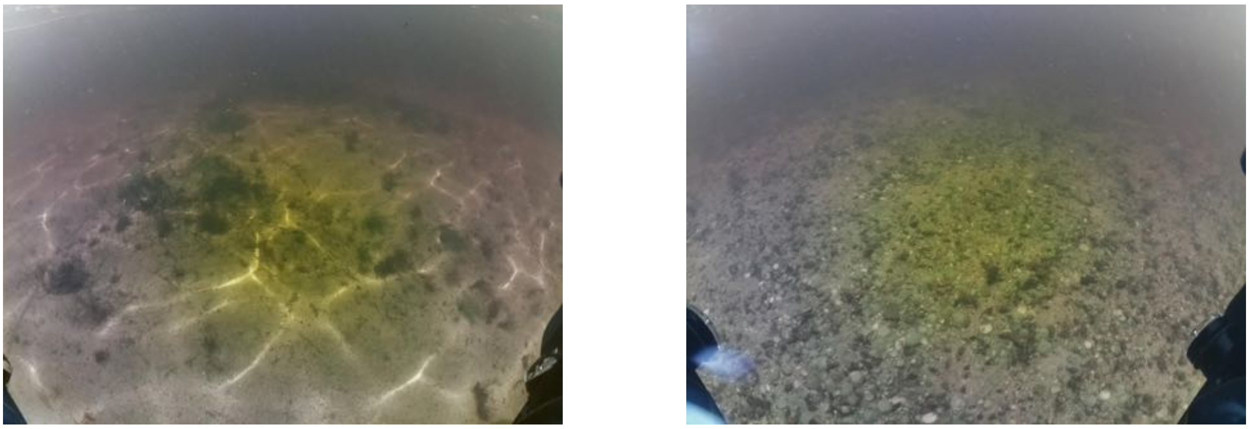
(b)

Figure 6.7: ORB-SLAM2 results for pool experiment. (a) without, (b) with markers

Point Walter—Swan River at Point Walter, Bicton, Perth

Description: The first three datasets: *Sunny_along_jetty*, *Sunny_circle* and *Sunny_rectangle* were recorded during a sunny day. The other three: *Cloudy_circle*, *Cloudy_eight* and *Cloudy_rectangle*, with cloudy weather. The different weather conditions allowed observation of the influence different lighting conditions have on ORB-SLAM2's results.

The surrounding at Point Walter is a sandy sea floor with a few rocks and some algae. In some images the foundations of the jetty from which the ROV was launched are visible as well. It is obvious that Fig. 6.8 (a) was taken on a sunny day as there is a lot of light ripples on the sea floor. In Fig. 6.8 (b), on the other hand, these ripples are not present.



(a) Sunshine

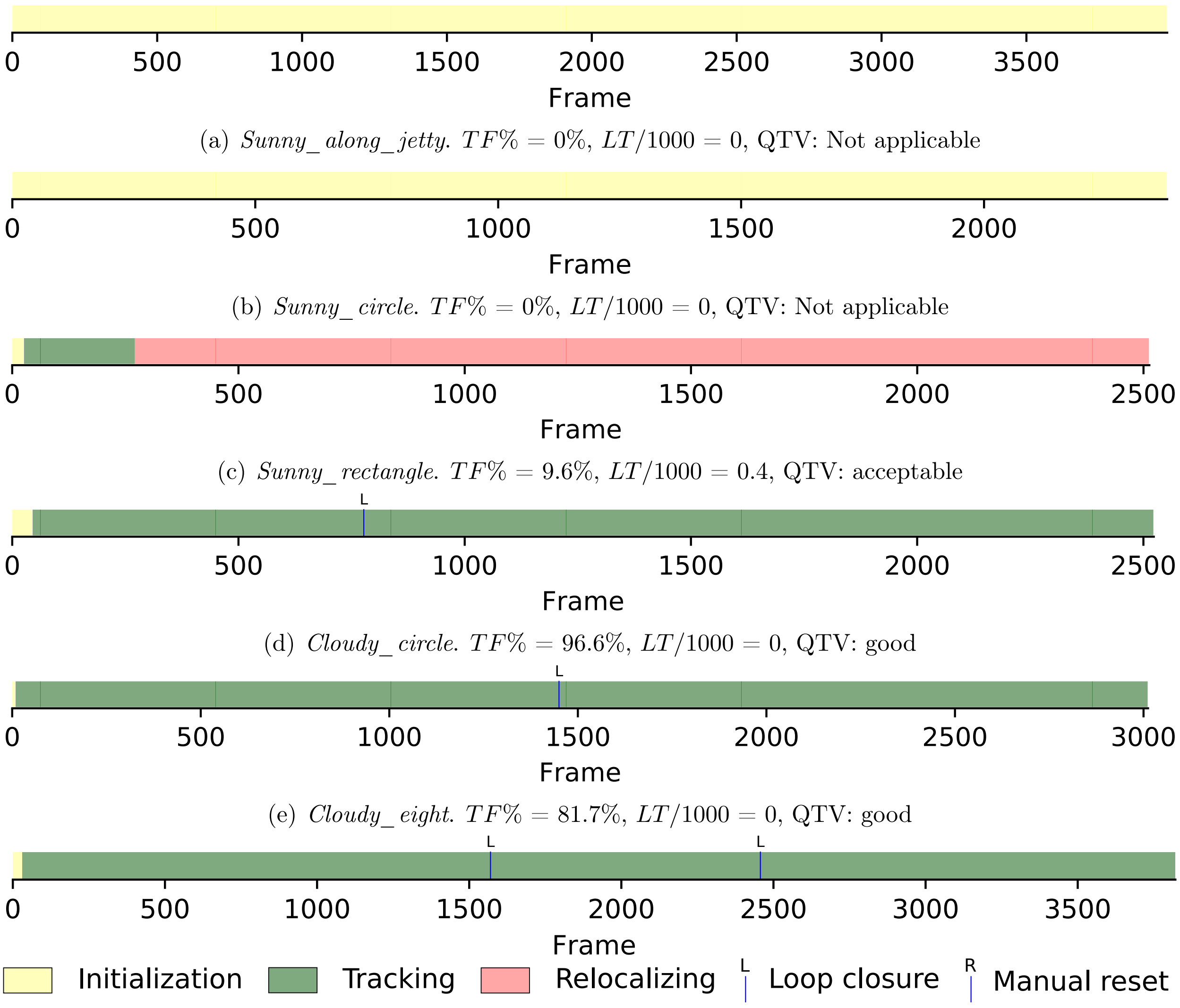
(b) Cloudy weather

Figure 6.8: Point Walter experiment: sample images

Evaluation: Looking at the plots in Fig. 6.9 it is evident that ORB-SLAM2 struggles when faced with the lighting conditions on a sunny day. Due to the ripples visible in Fig. 6.8(a), ORB-SLAM2 cannot initialize and has no chance of tracking the robot's movement. In fact, for *Sunny_rectangle* it is only able to initialize because at the beginning of the recording there is a cloud in front of the sun, changing the lighting conditions for a short time. As soon as the cloud is gone and the ripples are seen again, ORB-SLAM2 loses tracking and can neither relocalize nor reinitialize when the algorithm is reset.

In contrast, tracking works very well for datasets collected during a cloudy day. As presented in Fig. 6.10, ORB-SLAM2 is able to recover the driven trajectory accurately. The fact that the shapes are not perfect does not stem from ORB-SLAM2 not tracking the driven trajectory correctly, but rather from not being able to drive the same trajectory flawlessly multiple times. An interesting observation that can be made from the results on these datasets is that ORB-

SLAM2 detected very little loop closures. This does not mean that the algorithm fails at detecting possible loop closures, but rather that it is able to recognize already mapped places, even in an environment as monotonous as the one at Point Walter.



(a) *Sunny_along_jetty*. $TF\% = 0\%$, $LT/1000 = 0$, QTV: Not applicable

(b) *Sunny_circle*. $TF\% = 0\%$, $LT/1000 = 0$, QTV: Not applicable

(c) *Sunny_rectangle*. $TF\% = 9.6\%$, $LT/1000 = 0.4$, QTV: acceptable

(d) *Cloudy_circle*. $TF\% = 96.6\%$, $LT/1000 = 0$, QTV: good

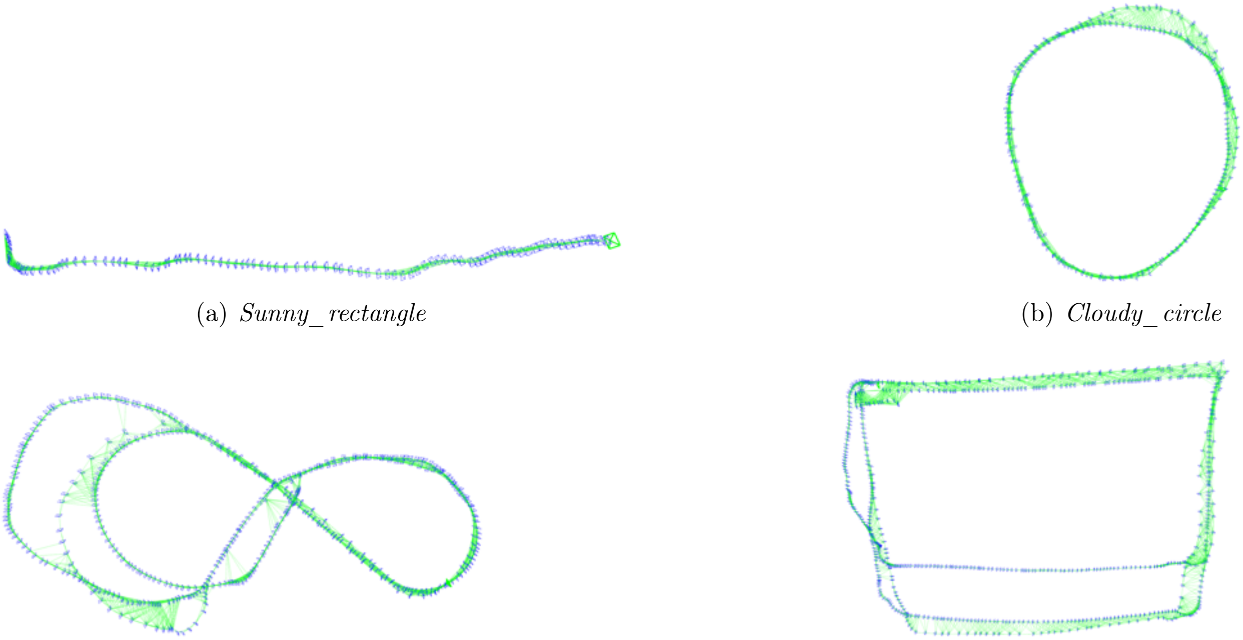
(e) *Cloudy_eight*. $TF\% = 81.7\%$, $LT/1000 = 0$, QTV: good

(f) *Cloudy_rectangle*. $TF\% = 99\%$, $LT/1000 = 0$, QTV: good

Figure 6.9: Point Walter experiment results

Fremantle Marina

Description: This area was right at the sea, but surrounded by stone walls so there are barely any waves. The point where the robot was put in the water was characterized by a steep slope made of loose stones which leads down to a flat sandy ground with a few plants and some man-made objects like pipes and a chair.



(a) *Sunny_rectangle*

(b) *Cloudy_circle*

(c) *Cloudy_eight*

(d) *Cloudy_rectangle*

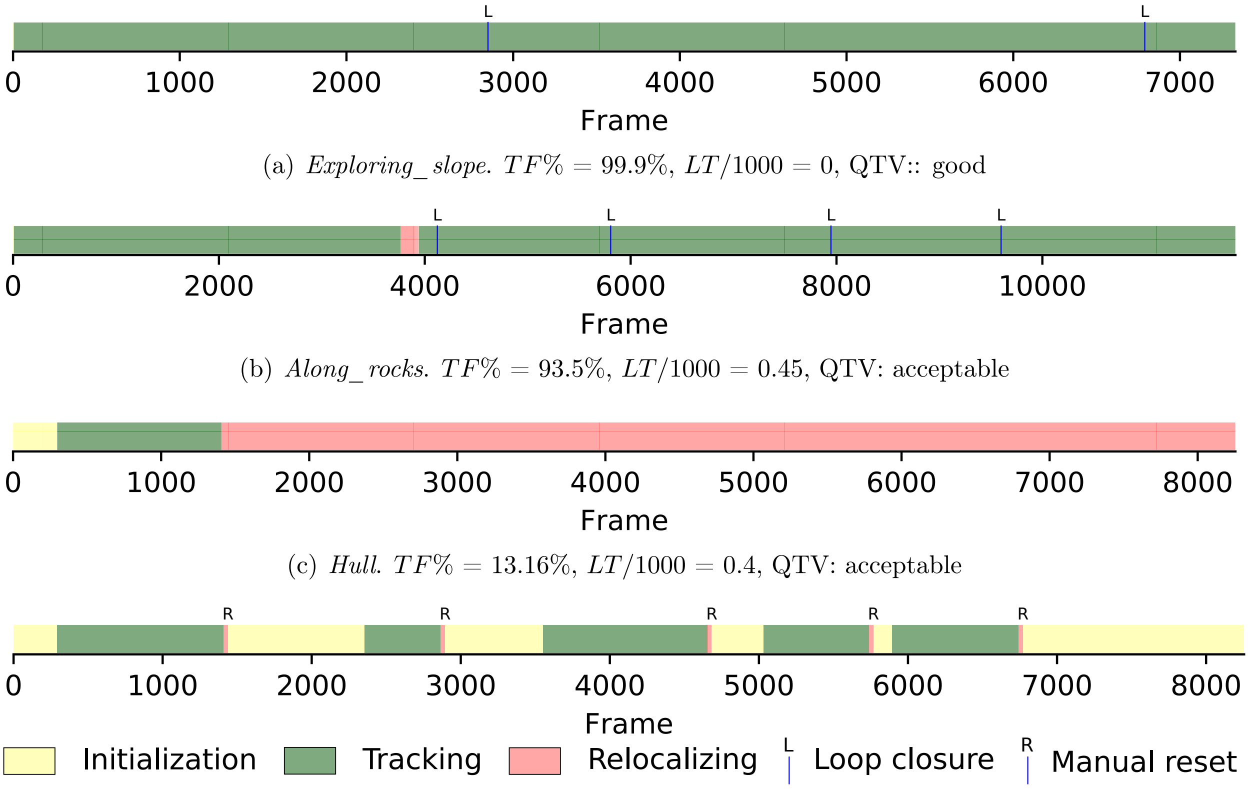
Figure 6.10: ORB-SLAM2 trajectory estimates for Point Walter experiments

So far, all described experiments had been performed at very shallow areas with the ROV at the water's surface or just beneath it. The goal of the experiment *Exploring_slope* was to test how ORB-SLAM2 would work when varying the depth and moving to deeper areas. The slope at this spot allowed to follow it to a depth of about three meters. For this experiment the ROV was moved around this area in wide circles while moving up and down the slope.

The second and third datasets (*Along_rocks* and *Hull*) were recorded at nighttime. In this area there are a lot of submerged objects like a shopping cart, a tyre and two large pipes. The ROV was driven back and forth along the slope so it could observe the objects from different viewing angles.

For the *Hull* dataset the ROV was driven along the side of the hull of a boat anchored in the marina. This was done to see whether ORB-SLAM2 is able to deal with the ROV's camera facing upwards and no other visible surroundings but the submerged part of a boat as shown in Fig. 6.13 (c).

Evaluation: As is visible in Fig. 6.11 (a) tracking works perfectly for the first dataset. ORB-SLAM2 is able to track every single frame and never gets lost in any of the 10 test runs. Fig. 6.12 (a) shows the estimated trajectory from above a few frames before the algorithm detects a loop. As can be seen there it is able to track the ROV's movement back to the starting point without much error and it is able to do this over a long path with varying depth.



(a) *Exploring_slope*. $TF\% = 99.9\%$, $LT/1000 = 0$, QTV:: good

(b) *Along_rocks*. $TF\% = 93.5\%$, $LT/1000 = 0.45$, QTV: acceptable

(c) *Hull*. $TF\% = 13.16\%$, $LT/1000 = 0.4$, QTV: acceptable

(d) *Hull* with resetting. $TF\% = 55.2\%$, $LT/1000 = 1.16$, QTV: acceptable

Figure 6.11: Fremantle Marina experiment results

Experiment *Along_rocks* validates that ORB-SLAM2 is also able to track the robot's movement at night. But, even though it was possible to track the ROV for almost all frames ($TF\% = 93.5\%$), there is a major flaw within its result— after the ROV has reached the point furthest away from the start and turns around, ORB-SLAM2 does not reuse the map already created but rather creates a complete second map of the same environment. This effect is visible in Fig. 6.12 (c) where there are two representations of the pipe in the created map. As soon as the first loop is closed the drift is corrected as shown in Fig. 6.12 (d). ORB-SLAM2 however does not remove the duplicate points, but uses one part of the map for going in one direction and the other for the other direction. With regard to the *Hull* experiment, ORB-SLAM2 looses tracking very quickly. This happens because the lights on the BlueROV2 cannot be turned up far enough which leads to only small parts of the hull being visible as shown in Fig. 6.13 (c). Because ORB-SLAM2 is not able to relocalize for the remainder of the experiment, it is repeated with automatic resetting after 30 frames of failed relocalization. As is visible by the difference between Fig. 6.11 (c) and (d) it misses out on a lot of tracking opportunities by permanently trying to relocalize.

Another problem that was apparent during this experiment is that when the boat moves, ORB-SLAM2 mirrors this movement onto the ROV. In Fig. 6.12 (d), part of the mapped hull is shown.

Omeo Wreck, Coogee Beach, Perth

Description: The Omeo was a trading ship that sunk in 1905 close to the shores of Perth [227]. Since the wreck lies only about 20 meters form shore it was very easy to access. For this experiment, the ROV was driven up and down the length of the wreck. The footage contains a mix of structured and unstructured surroundings as the wreck lies on sandy ground.

Evaluation: As is visible in Fig. 6.14(a), ORB-SLAM2 loses track of the robot's movements. This is mostly due to the described mix of unstructured and structured areas present in the dataset. While the ROV moves around structured areas tracking works well and the estimated trajectory closely follows the robot's motion. When moving over unstructured areas tracking quickly fails because it cannot find enough features. This problem is demonstrated in Fig. 6.15.

Even though the data offers a lot of opportunities for loop closing and relocalization, ORB-SLAM2 is not able to pick up on most of them. For some this most likely happens because the change in viewing angle is too large, but in other cases the algorithm does not pick up on the possibility although the viewing angle is similar. This circumstance leads to tracking mostly being lost early on with only a few runs being able to track over a longer time.

When resetting the algorithm after 30 frames instead of waiting for relocalization the overall tracked frames can be increased a lot ($TF\% = 90\%$). The bar plot can be seen in Fig. 6.14 (b). This also allows for the detection of loop closures which highly improves the estimated trajectory. The result is shown in Fig. 6.16.

Open Sea, West of Fremantle, Perth

Description: The robot was launched from a boat directly into the sea. The goal of this experiment was to test how ORB-SLAM2 would react to all the conditions that come with working in the open sea. These datasets included heavy movements due to strong surge, lots of marine life, long algae which moves with the water and, since it was recorded on a sunny day, very dynamic lighting.

Dataset *Reef* incorporated many of these problematic conditions. In this dataset the ROV started at a depth of about one meter and slowly moved down to 5.5 meters while moving along large rocks covered in algae and coral. Since the robot started close to the surface there were rays of light moving through the water for the first half of the dataset. Furthermore, there was a lot of wildlife at this spot with schools of fish circling the robot, as can be seen in Fig. 6.17 (a). Since there was a strong surge the algae on the rocks moves a lot.

In dataset *Poles* the robot started at the ground at about 5.5 meters in an area with three wooden poles covered in coral which stretch from the bottom of the sea to above surface level. Here the ROV moved along a piece of wood which lies on the sea bed to where the three poles were and then moved along the poles standing upright. After this, the ROV followed a pole all the way from the floor to the surface and back down before it leaves the scene along the piece of wood where it began. This area also contained quite a bit of movement since there was heavy surge, fish and moving algae.

Evaluation: Fig. 6.18 (a) demonstrates that ORB-SLAM2 cannot handle the highly dynamic environment the *Reef* dataset was recorded in. Most tests completely fail to track even a single frame ($TF\%$ <1).

It is worth noting that the problematic effects of sunlight weaken with depth is further reinforced by dataset *Poles*. In this dataset the robot starts at a depth of about 5.5 meters and shows no trouble with initializing, even though the data was recorded on the same day with only a short boat ride between them.

Overall, the results produced on this dataset are much better in comparison to the reef dataset. In this case, the $TF\%$ achieved is almost 97%. There is, however, also a significant amount of tests where tracking is lost due to a quick movement (i.e. around frame 2000), as shown in Fig. 6.18(c). As is also the case for datasets *Along_ rocks* in Fremantle and *Along_ wreck*, in the Omeo Wreck, the $TF\%$ and therefore, the overall gathered information, can be increased by resetting the algorithm when it tries to relocalize for too long.

Due to its three-dimensionality this is a dataset for which it is very easy to follow whether ORB-SLAM2's estimation is accurate or not. The results are presented in Fig. 6.19. As can be seen there the algorithm is able to create an accurate estimation of the three vertical poles and how the robot moved around them.

6.5 Results

6.5.1 Monotony

The problem of having areas with minimal structural change is observable multiple times within the executed experiments. Especially, the datasets from Point Walter and Omeo Wreck contain a lot of monotonous areas. What is interesting about these datasets is that the results on the two different locations vary a lot. The experiments show that monotonous surroundings can be an issue for ORB-SLAM2's tracking, but it only really becomes a problem when the areas are both monotonous and low in texture. While tracking gets lost often due to low texture areas in the Omeo Wreck experiment, the experiment at Point Walter proved to have one of the best overall results. The reason for this is depicted in Fig. 6.20. As is evident ORB-SLAM2 is not able to extract enough features in low texture areas, Fig. 6.20 (a), as there are little corners. As soon as there are a few spots of different color on the ground as in Fig. 6.20 (b) the algorithm is able to extract more features which helps with tracking.

6.5.2 Turbidity

Having particles flowing in the water is something that can be observed in every single dataset collected. In practice ORB-SLAM2 has demonstrated that this is mostly not a problem. Fig. 6.21 demonstrates this, as can be seen in the upper right corner of Fig. 6.21 (a), ORB-SLAM2 does extract features from particles floating on the top. Fig. 6.21 (b) however shows that none of these features were considered a match.

Even though ORB-SLAM2 does extract features from particles, in most cases these are not matched. This happens mostly due to the fact that features extracted from particles move differently between frames than the features on the ground do. In spite of proving that good results can be achieved in the presence of turbidity, it was not possible to rule out that they have an influence on the outcome.

6.5.3 Dynamics

The dynamical nature of the underwater environment is most noticeable in those datasets recorded in open sea. There are fish swimming through the images, algae moving with the surge and strong currents and waves moving the robot in unexpected ways. The impact of these dynamic changes on ORB-SLAM2 is most apparent during initialization (Fig. 6.18 (a)). As shown in Fig. 6.22 the algorithm tends to produce false feature matches during initialization when there is a lot of motion in the scene. This often leads to a repeatedly failing initialization. In Fig. 6.22, green lines connect feature matches between the initial and the current frame. In normal cases the green lines produced align with the motion of the robot. As in dynamic environments the features extracted move independent from how the robot moves ORB-SLAM2 creates wrong matches and often fails to initialize.

Nevertheless, once the algorithm is able to initialize, the dynamic movements do not seem to have an effect on the tracking result anymore. Much like it is the case for turbidity the movement causes the features to move too much which results in ORB-SLAM2 not tracking them. Fig. 6.23 shows an image of moving algae, in the presence of moving objects ORB-SLAM2 is not able to match features extracted within these objects. Instead it only tracks points which are in areas with little motion as can be seen in this image where most tracked features lie between the moving algae.

But, even if there is no direct observable influence on the quality of tracking it can easily cause tracking to fail due to not being able to find enough matches.

6.5.4 Lighting

Lighting is definitely the environmental influence with the biggest impact on the results of ORB-SLAM2. This is especially apparent when looking at the results of the Point Walter experiments. The datasets from Point Walter were recorded on two different days with different lighting conditions. While ORB-SLAM2 cannot even initialize (Fig. 6.9 (a), (b)) in the datasets during a sunny day, the results on the datasets with cloudy weather proved to be some of the best overall. The problems on datasets with a lot of sunlight stem from the light ripples on the ground shown in Fig. 6.8 (a). As can be seen in Fig. 6.24, the ripples on the floor move a lot even in the short time between two images (less than 100 ms) which makes the scene very dynamic and keeps ORB-SLAM2 from matching the features. These ripples will ORB-SLAM2 to detect many features along them.

Furthermore, the lighting also causes reflections in the ROV's casing as in the upper left corner of the images above. It also makes particles in the water more reflective and leads to even more features extracted on these. All these effects combined make the scene so highly dynamic that initialization constantly fails.

The negative effects of sunlight decrease when moving to deeper depth as shown in the datasets from the sea. Once the ROV moves below 3-4 meters the ripples are less noticeable and initialization is more likely to be successful.

6.6 Chapter Summary

The evaluation performed on over 40 different datasets in varying environments showed that ORB-SLAM2 performance is satisfactory given scenarios with sparse features and low light flickering. However, it also shows that the algorithm struggles with some of the characteristics of the underwater environment such as highly dynamic lighting and surroundings with lots of moving objects like fish and algae.

Light flickers and turbidity overlay real features and move rapidly creating inconsistent ORB features among consecutive frames causing the algorithm to lose track or never initialize.

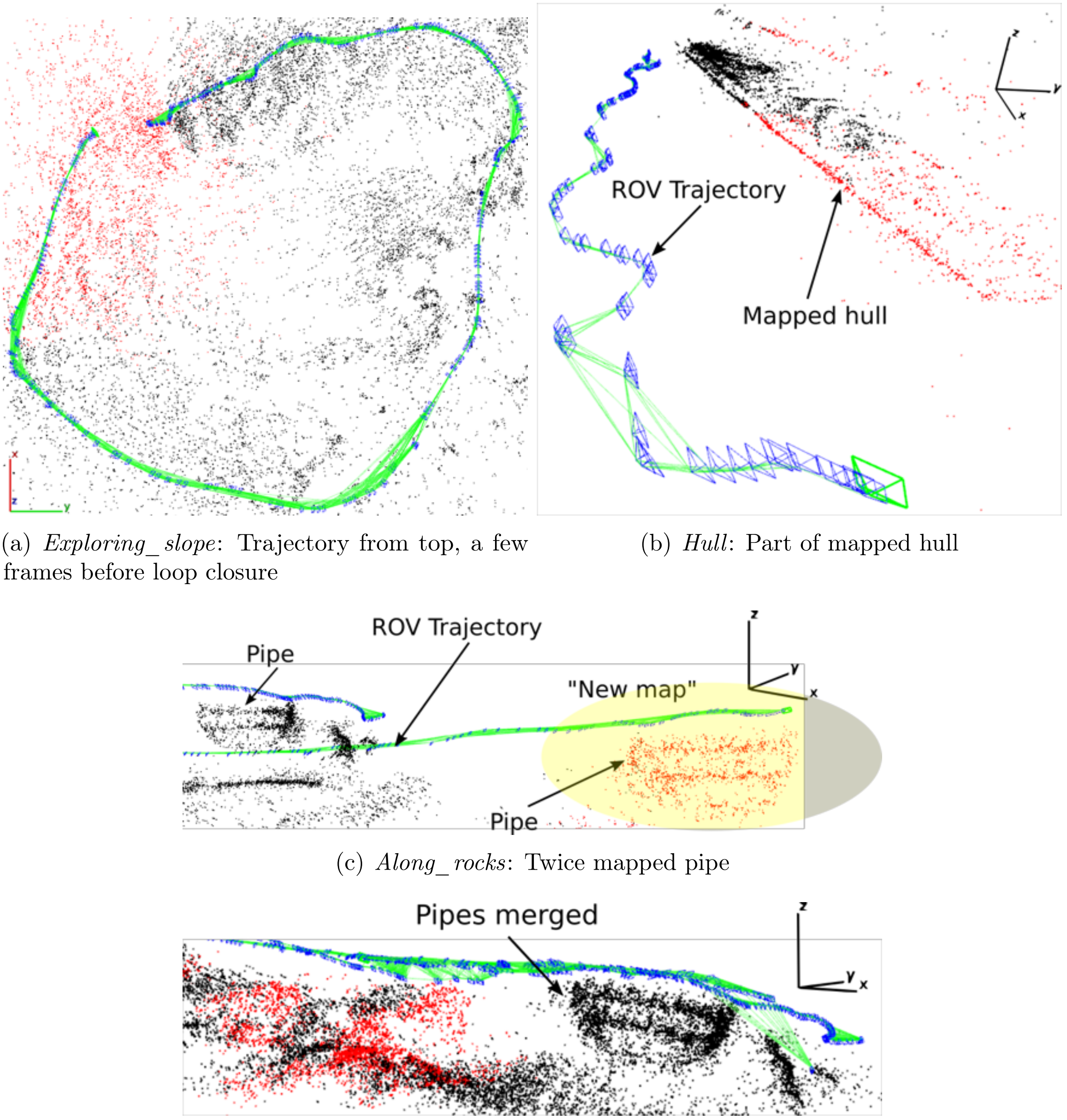
Different image processing can be used to reduce degradation effects in underwater images [228] such as filtering and contrast enhancement [229][230]. A polarizer can also be used to reduce illumination problems [231]. Recent approaches like [232] or [233] were able to show that using illumination invariant image transformations as proposed in [234] could greatly improve on the problem introduced by varying lighting conditions.

Low-texture areas as sand patches are difficult for ORB-SLAM to find relevant features. A very interesting idea for overcoming this would be to mix feature-based methods with dense approaches as in DTAM and Distributed Particle (DP)-SLAM [235] based on visible gradients.

Fast large moving objects and fast movements of the robot cause the algorithm to lose tracking. After this, the algorithm tries to relocalize and ignores a lot of new information. To overcome this, whenever tracking is lost two threads could be started, one thread that tries to relocalize within already known maps and another thread that starts the normal initialization and tracking process. When initialization is successful a new map is created which can be merged with existing maps once the relocalization thread finds a match within an older map.

As already proposed by the ORB-SLAM2 authors in [236] introducing the usage of Inertial Measurement Unit (IMU) measurements into the algorithm can overcome some of the problems linked to monocular SLAM such as scale drift while simultaneously estimating gyroscope and accelerometer biases. Furthermore, it also allows to extract direction of gravity which allows to create maps aligned with the actual environment.

The results presented in this chapter can be used as a baseline to compare future improvements of the algorithm. As can be seen from our results, there is still a lot of room for improvement for ORB-SLAM2 in the underwater environment. Especially filtering the images, to overcome lighting problems; combining feature-based with dense approaches, to handle monotonous areas and; including motion or positioning sensors, to improve heading, scaling and loss of tracking; could greatly improve the results.



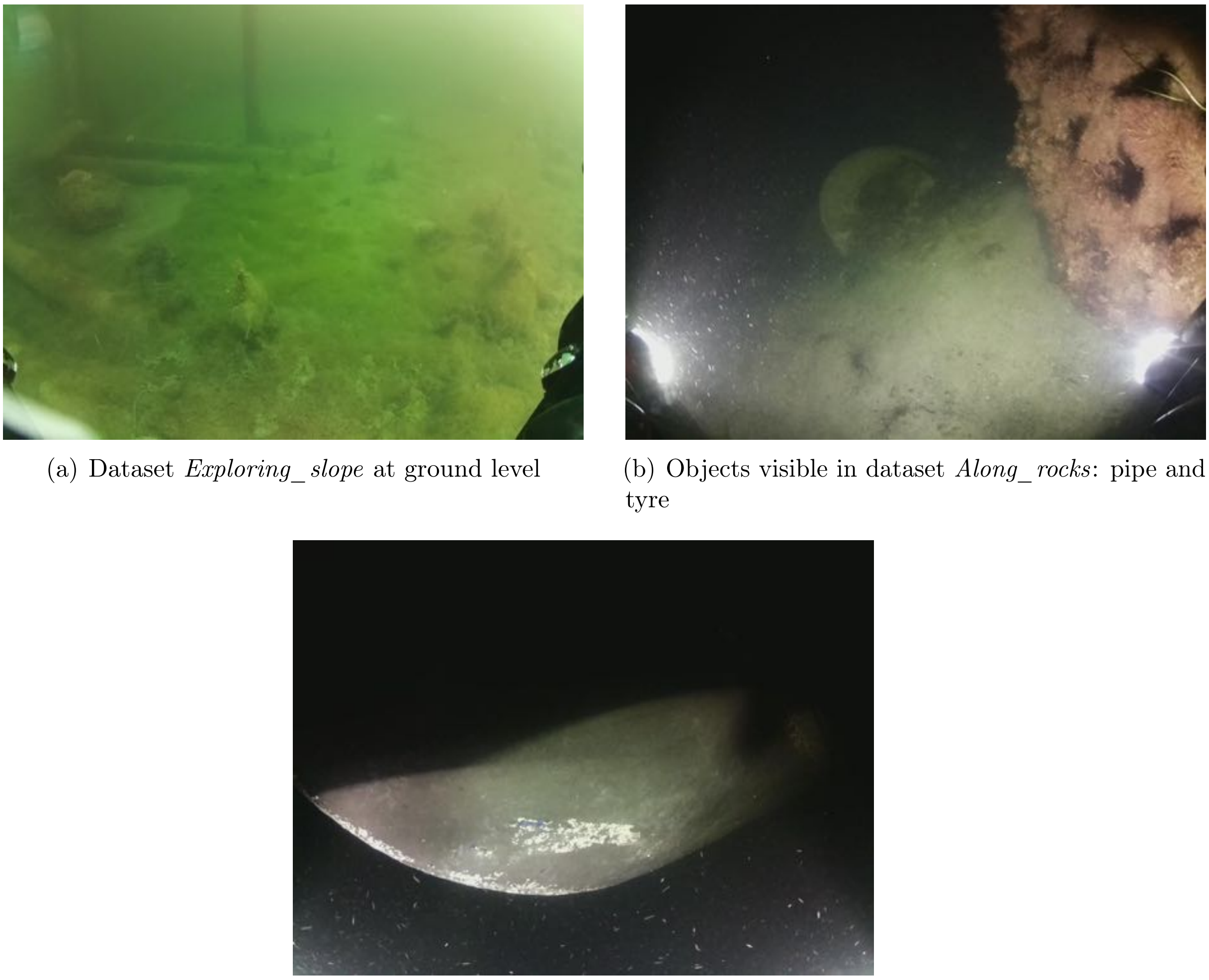
(a) *Exploring_ slope*: Trajectory from top, a few frames before loop closure

(b) *Hull*: Part of mapped hull

(c) *Along_ rocks*: Twice mapped pipe

(d) *Along_ rocks*: Merged pipes after loop

Figure 6.12: ORB-SLAM2 results for Fremantle Marina experiments

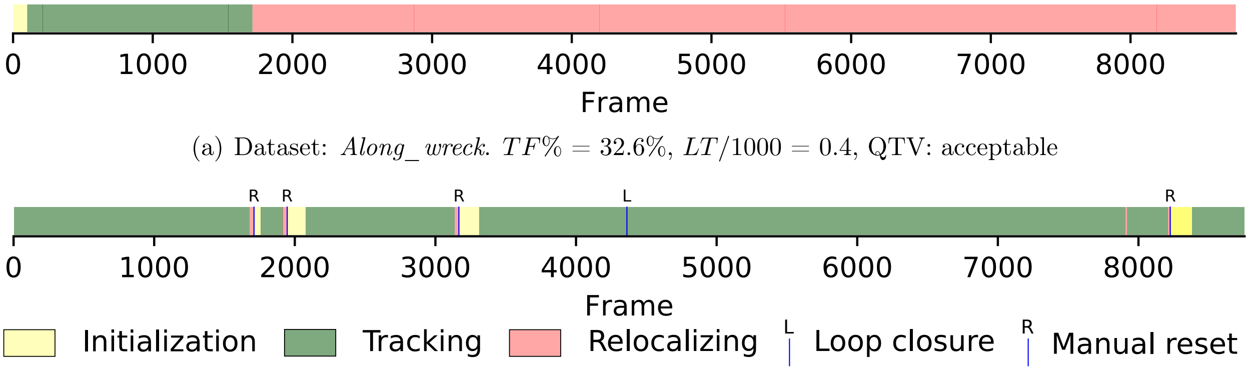


(a) Dataset *Exploring_slope* at ground level

(b) Objects visible in dataset *Along_rocks*: pipe and tyre

(c) Boat hull at night, dataset *Hull*

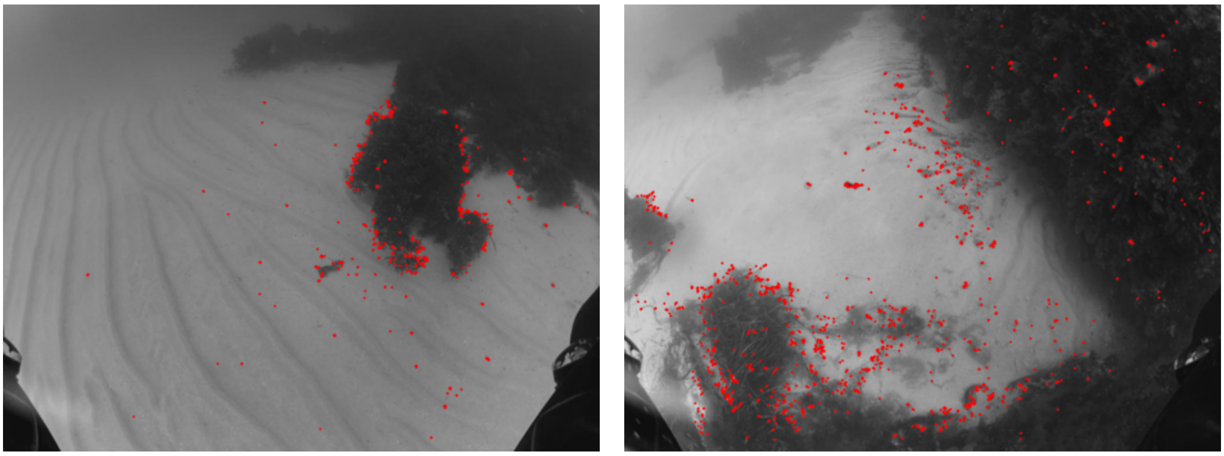
Figure 6.13: Fremantle Marina experiment sample images



(a) Dataset: *Along_wreck*. $TF\% = 32.6\%$, $LT/1000 = 0.4$, QTV: acceptable

(b) Dataset: *Along_wreck* with resetting. $TF\% = 90.1\%$, $LT/1000 = 0.8$, QTV: acceptable

Figure 6.14: Omeo wreck experiment results



(a) Low number of features found, mostly grouped in a smal area

(b) High number of features found, good distribution in the frame

Figure 6.15: Images showing problematic feature detection on areas consisting mainly of sand. While ORB-SLAM2 can find a lot of features in structured areas it is barely able to find any on sandy ground.

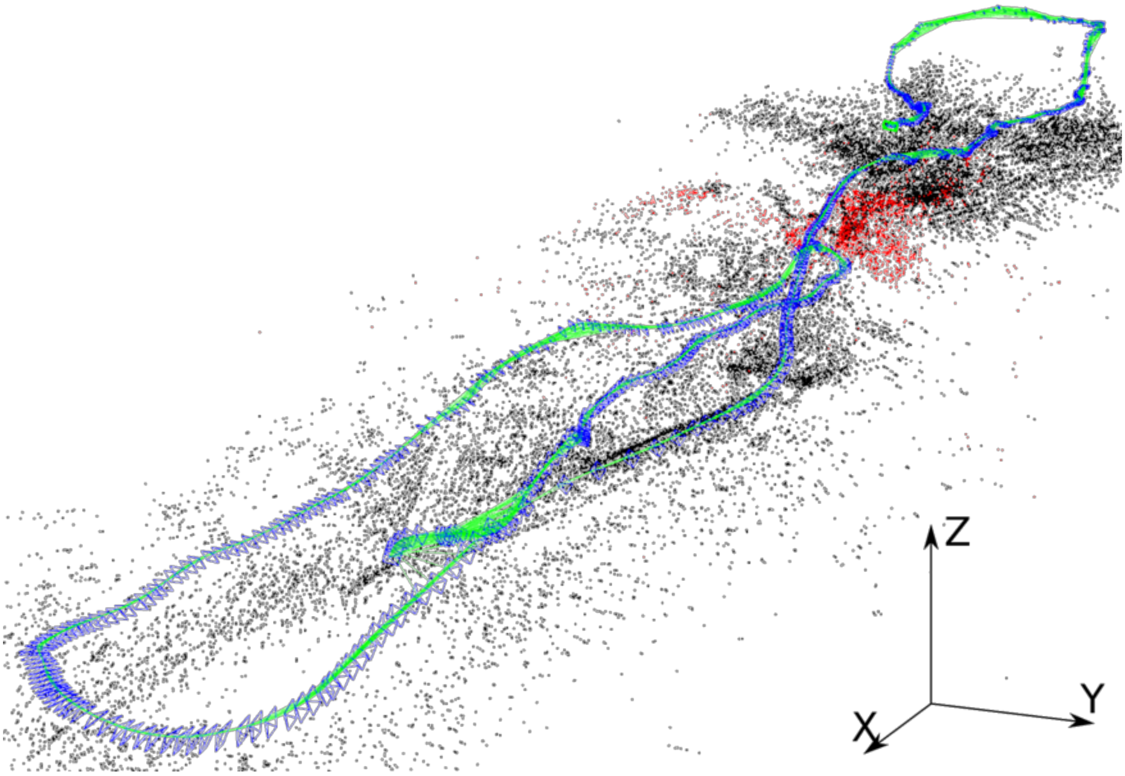
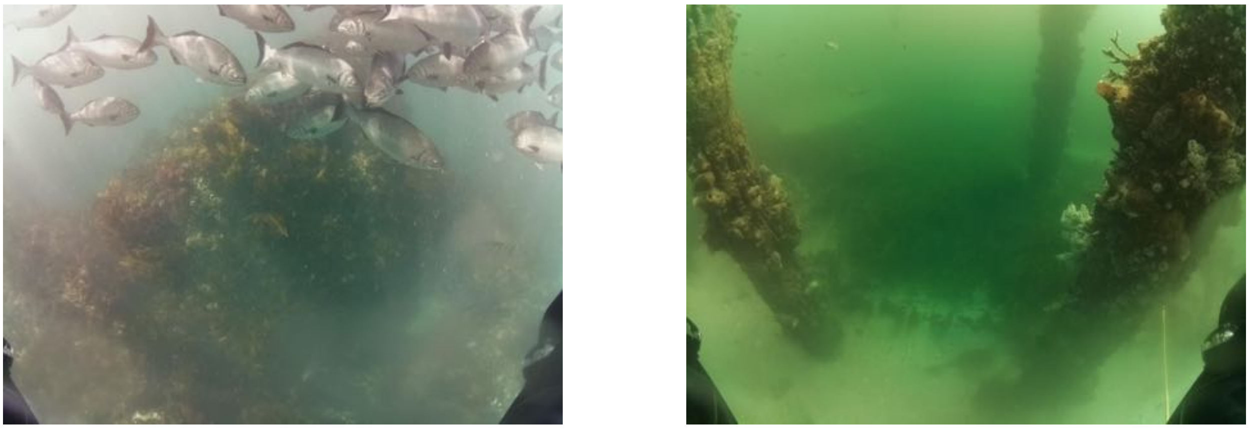


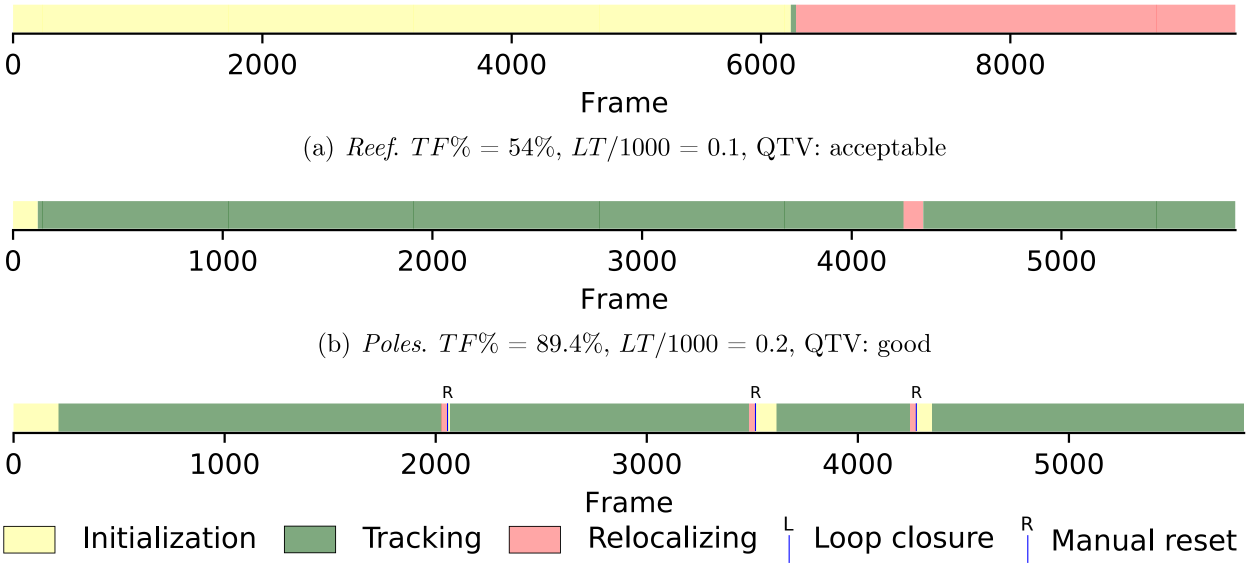
Figure 6.16: ORB-SLAM2 results for Omeo Wreck experiment



(a) *Reef*

(b) *Poles*

Figure 6.17: Boat experiment sample images



(a) *Reef*. $TF\% = 54\%$, $LT/1000 = 0.1$, QTV: acceptable

(b) *Poles*. $TF\% = 89.4\%$, $LT/1000 = 0.2$, QTV: good

(c) *Poles* with resetting. $TF\% = 92.3\%$, $LT/1000 = 0.5$, QTV: good

Figure 6.18: Boat experiment results

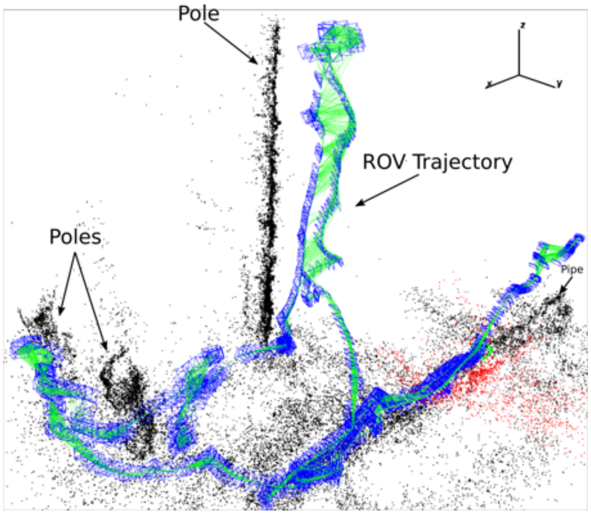
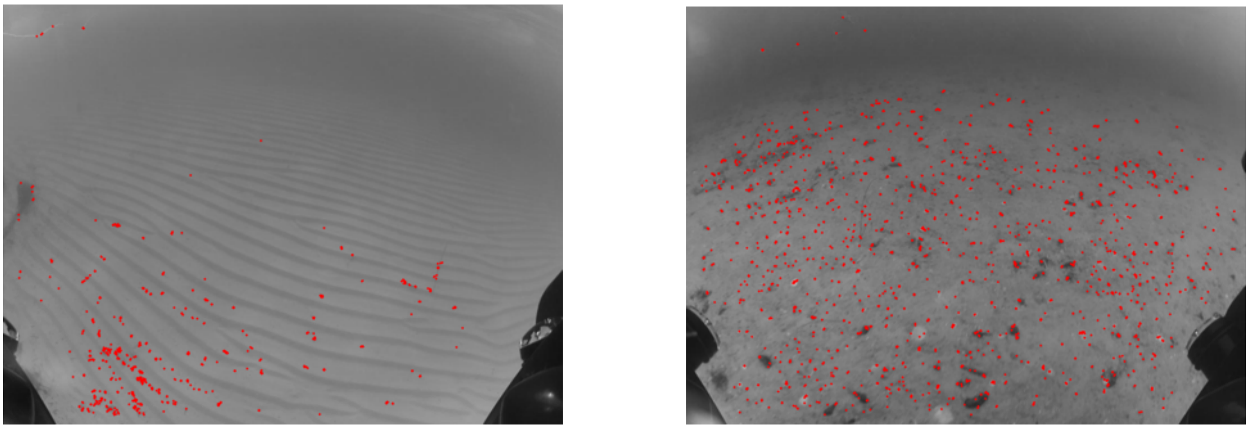


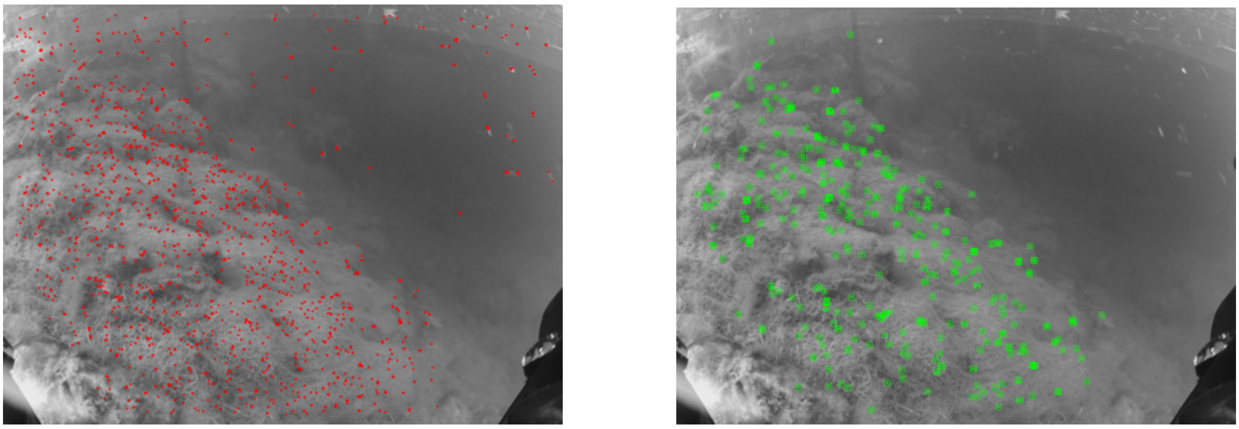
Figure 6.19: ORB-SLAM2 results for Boat experiment



(a) Low texture area in Omeo Wreck experiment

(b) High texture area in Point Walter experiment

Figure 6.20: A comparison of extracted features in monotonous areas with low (a) and high texture (b). The red dots resemble extracted corners



(a) Features extracted

(b) Matches found

Figure 6.21: A frame showing algae covered rocks taken from dataset *exploring_slope*. (a) shows the extracted features in red. (b) illustrates matches found between the last frame's and the current frame's features in green.

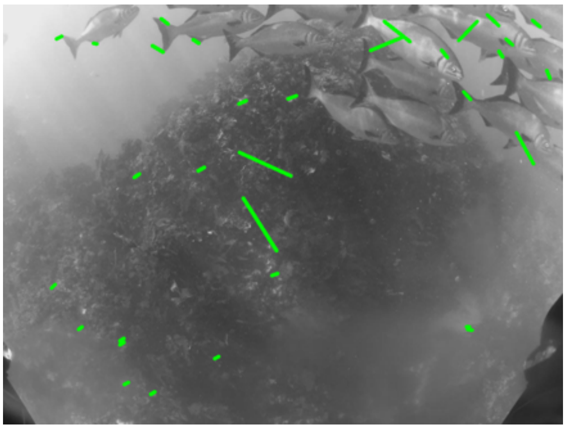


Figure 6.22: Initialization in a highly dynamic environment

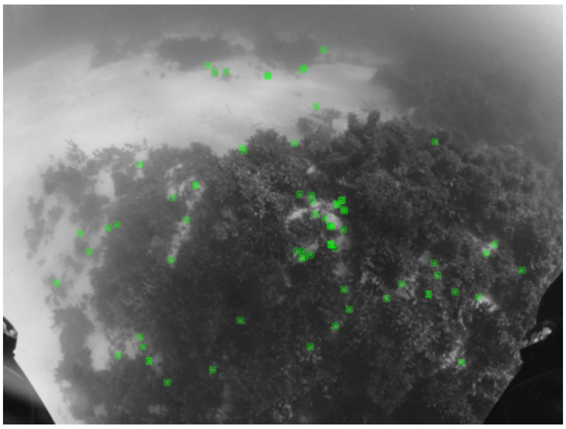
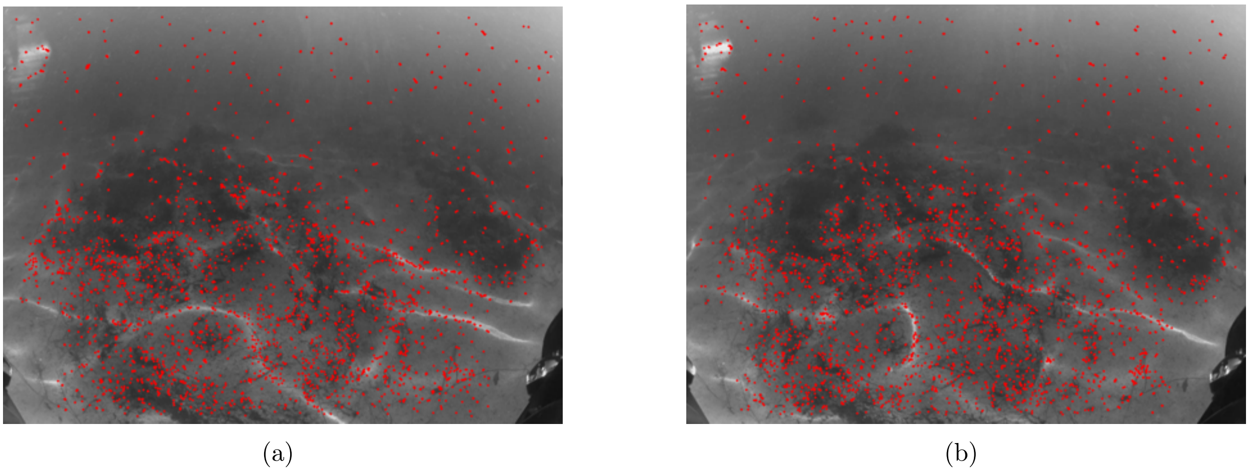


Figure 6.23: Tracking in a scene with moving algae



(a)

(b)

Figure 6.24: Feature extraction on two consecutive images of dataset *Rectangle* from Point Walter

Chapter 7

Conclusions

7.1 Summary

This thesis presented research and developments in underwater robots implementation and in Simultaneous Localization and Mapping (SLAM) applied to marine scenarios. Instrumentation and hardware were reviewed into the implementation of an Remotely Operated Vehicle (ROV) for scientific environmental research as well as into a development framework for robots. Furthermore, different approaches to SLAM applied to underwater robots were reviewed and compared. Special emphasis to vSLAM was given to exploit advances in computer vision related to feature detectors and descriptors.

The first part of this thesis is related to underwater robot hardware. In Chapter 2, we built and tested an ROV-based acquisition system for scientific research based on the studies performed by the Instituto del Mar del Perú (IMARPE). We proposed the integration of sensors into a single platform to record data for scientific research. In Chapter 3, we introduced a robotics control framework based on a low-level interface board and a set of software libraries applicable to underwater robots, and we used it to upgrade two AUVs. The framework proved to be successfully integrated into underwater robots and allowed a modular control structure.

The second part of this dissertation focused on SLAM in underwater robots. In Chapter 4, we reviewed, and evaluated three main frameworks for a simulated dataset. We showed how the separation of SLAM into front-end and back-end allows processing a single dataset into three different algorithms, as well as the improvements in the estimation of the robot's pose and detected landmarks. A table that compiles selected works in underwater SLAM was also presented. In Chapter 5, we started to focus on vSLAM, by evaluating different feature detectors and descriptors in their application to vSLAM for underwater environments. We collected and made public a collection of underwater video footage in a variety of conditions.

The evaluation showed the suitability of all the detectors and descriptors to vSLAM, resulting in Oriented FAST and Rotated BRIEF (ORB) being the less computationally expensive with one of the best performances in matching features in consecutive frames. In Chapter 6, we applied the vSLAM approach ORB-SLAM2 into the collected datasets and evaluated its performance. It showed acceptable performance in uniform light conditions but, struggled against the light caustic pattern and non-uniform illumination, as well as in low textured backgrounds such as sandy areas.

7.2 Findings

This research results in four main conclusions:

- Different mobile robot frameworks can be used to develop underwater robots. The one proposed in Chapter 3 worked successfully in the upgrade of two AUVs and it allows further development and integration to higher levels of control. (Chapters 2 and 3)
- SLAM algorithms applied to underwater robots improves the estimation of the robot pose. The front-ends and back-ends from different approaches can be adapted to be interchangeable, being the graphSLAM back-end a promising approach especially for their suitability to vSLAM. (Chapter 4)
- Algae, sand patches, rocks, debris, and poles show identifiable features to Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), ORB, Binary Robust Invariant Scalable Keypoints (BRISK) and Accelerated-KAZE (AKAZE) extractors which allows the application of vSLAM in underwater environments. (Chapter 5)
- Underwater images altered by illumination effects such as non-uniform illumination and caustic patterns showed challenges at the time of matching features among frames hindering the application of vSLAM. Underwater enhancement algorithms improve the performance of feature detectors in these scenarios. (Chapters 5 and 6)

7.3 Future Work and Open Problems

The hardware and software implementations in Chapters 2 and 3 are a gateway to different applications and further developments. Preliminary tests were done on the ROV for scientific research, but the final goal for it is to be incorporated as a regular tool for the regular surveys conducted by the IMARPE. Proper methodology and validation have to be completed first in cooperation with the developers. Additionally, the sensors such as the Inertial Measurement Unit (IMU) and the depth sensor can be exploited in assisted navigation (constant depth, orientation, and both) as the BlueROV2. Localization is essential in data collection, therefore, it is necessary to implement sensor fusion as well as the SLAM applications described in Chapters 4 and 6 for geo-referencing.

Similarly to the ROV, the AUVs implemented using the Eyebot platform in Chapter 3 offers a variety of future development options towards autonomous navigation. There are many packages for robotics applications available for Robot Operating System (ROS) that can be exploited easily thanks to the ROS integration that Eyebot offers and the simulation platform. A future integration of the EyeSim VR with ROS is also planed.

SLAM based on graph optimization is getting popular especially in visual SLAM (vSLAM). The application of monocular ORB-SLAM2 in underwater environments presented in Chapter 6 showed an acceptable performance as a baseline for future development. In the case of the use of monocular cameras, the robot trajectory and the map need to be scaled afterward since they do not capture depth and the 3D abstraction is based on an initialization module. The integration of conventional onboard sensors such as the IMU and a pressure sensor into the vSLAM algorithm is advisable. In Chapter 4 we showed the use of Kalman Filter (KF) and Particle Filter (PF) approaches to fuse sensor information to improve the accuracy of the pose estimation. These elements can be incorporated in a more complex SLAM approach exploiting the heading provided by the IMU and the almost absolute depth obtained by the pressure sensor.

The most salient problems found in Chapter 5, regarding the features detected in underwater images, are plain areas such as sandy seafloors and light alterations, especially in a caustic pattern. There is little to do to expose detectable features in sandy areas, although this situation improves when the robot captures images close to the sand the re-observation of them is not guaranteed. In the case of caustic patterns, the problem is that features get extracted from the patterns generated overlaying actual features from the scene. A couple of underwater image enhancement algorithms were tested demonstrating a slight improvement in the features extracted and matching features in consecutive frames.

Bibliography

[1] *Total Marine Technology.* URL: http://www.tmtrov.com.au/ (visited on 07/30/2018).

[2] *AC-CESS - Remotely Operated Vision And Sense.* URL: http://www.ac-cess.com/ (visited on 08/13/2018).

[3] *Teledyne Seabotix - Remotely Operated Vehicles (ROVs).* URL: http://www.teledynemarine.com/ (visited on 08/13/2018).

[4] *Ocean Modules Sweden AB - Remotely-Operated Vehicles (ROV) and SPOT.ON Survey Software.* URL: http://www.ocean-modules.com/ (visited on 08/13/2018).

[5] *Kongsberg Gruppen.* URL: https://kongsberg.com/ (visited on 07/30/2018).

[6] *Girona Underwater Vision And Robotics.* URL: http://cirs.udg.edu/ (visited on 08/13/2018).

[7] Frederico Oliveira et al. "Marine Litter in the Upper São Vicente Submarine Canyon (SW Portugal): Abundance, Distribution, Composition and Fauna Interactions". In: *Marine Pollution Bulletin* (2015). ISSN: 0025326X. DOI: 10.1016/j.marpolbul.2015.05.060.

[8] Kyra Schlining et al. "Debris in the Deep: Using a 22-Year Video Annotation Database to Survey Marine Litter in Monterey Canyon, Central California, USA". In: *Deep-Sea Research Part I: Oceanographic Research Papers* 79 (2013), pp. 96–105. ISSN: 09670637. DOI: 10.1016/j.dsr.2013.05.006.

[9] "Offshore Industry's Pollution". In: *Marine Pollution Bulletin* 18.2 (Feb. 1987), p. 58. ISSN: 0025326X. DOI: 10.1016/0025-326X(87)90549-2.

[10] L Bittencourt et al. "Underwater Noise Pollution in a Coastal Tropical Environment." In: *Marine pollution bulletin* 83.1 (June 2014), pp. 331–6. ISSN: 1879-3363. DOI: 10.1016/j.marpolbul.2014.04.026.

[11] Albert J. Williams. "Innovative Technology in Oceanography: Past, Present and Future". In: *2011 International Symposium on Ocean Electronics.* IEEE, Nov. 2011, pp. 3–17. ISBN: 978-1-4673-0266-1. DOI: 10.1109/SYMPOL.2011.6170491.

[12] Manhar Dhanak et al. “Using Small AUV for Oceanographic Measurements”. In: *Oceans ’99. MTS/IEEE. Riding the Crest into the 21st Century. Conference and Exhibition. Conference Proceedings (IEEE Cat. No.99CH37008)*. Vol. 3. IEEE & Marine Technol. Soc, 1999, pp. 1410–1417. ISBN: 0-7803-5628-4. DOI: 10.1109/OCEANS.1999.800200.

[13] Asher Bender, Stefan B. Williams, and Oscar Pizarro. “Autonomous Exploration of Large-Scale Benthic Environments”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 390–396. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6630605.

[14] Carol D C.D. Janzen and E.L. Creed. “Physical Oceanographic Data from Seaglider Trials in Stratified Coastal Waters Using a New Pumped Payload CTD”. In: *OCEANS 2011*. 2011, pp. 1–7.

[15] Kenneth M. Sharp and Randy H. White. “More Tools in the Toolbox: The Naval Oceanographic Office’s Remote Environmental Monitoring Units (REMUS) 6000 AUV”. In: *Oceans 2008* (Sept. 2008), pp. 1–4. DOI: 10.1109/OCEANS.2008.5152120.

[16] Amy L. Kukulya et al. “3D Real-Time Tracking, Following and Imaging of White Sharks with an Autonomous Underwater Vehicle”. In: *OCEANS 2015 - Genova*. IEEE, May 2015, pp. 1–6. ISBN: 978-1-4799-8736-8. DOI: 10.1109/OCEANS-Genova.2015.7271546.

[17] John E Elliott and Kyle H Elliott. “Environmental Science. Tracking Marine Pollution.” In: *Science (New York, N.Y.)* 340.6132 (May 2013), pp. 556–8. ISSN: 1095-9203. DOI: 10.1126/science.1235197.

[18] Jianhua Wan and Yang Cheng. “Remote Sensing Monitoring of Gulf of Mexico Oil Spill Using ENVISAT ASAR Images”. In: *2013 21st International Conference on Geoinformatics*. IEEE, June 2013, pp. 1–5. ISBN: 978-1-4673-6228-3. DOI: 10.1109/Geoinformatics.2013.6626165.

[19] Matthew Huelsenbeck and Caroline Wood. “Seismic Airgun Testing for Oil and Gas. A Deaf Whale Is A Dead Whale”. In: *OCEANA* (April 2013).

[20] Andrew M. Patterson, Jesse H. Spence, and Raymond W. Fischer. “Evaluation of Underwater Noise from Vessels and Marine Activities”. In: *2013 IEEE/OES Acoustics in Underwater Geosciences Symposium* (July 2013), pp. 1–9. DOI: 10.1109/RIOAcoustics.2013.6683985.

[21] Robert Kilpatrick et al. “Autonomous Video Camera System for Monitoring Impacts to Benthic Habitats from Demersal Fishing Gear, Including Longlines”. In: *Deep-Sea Research Part I: Oceanographic Research Papers* 58.4 (2011), pp. 486–491. ISSN: 09670637. DOI: 10.1016/j.dsr.2011.02.006.

[22] Kyung Woon Lee et al. "Implementation of Embedded System for Autonomous Buoy". In: *OCEANS 2011 IEEE - Spain*. IEEE, June 2011, pp. 1–4. ISBN: 978-1-4577-0086-6. DOI: 10.1109/Oceans-Spain.2011.6003447.

[23] Matthew Dunbabin et al. "A Hybrid AUV Design for Shallow Water Reef Navigation". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. April. IEEE, 2005, pp. 2105–2110. ISBN: 0-7803-8914-X. DOI: 10.1109/ROBOT.2005.1570424.

[24] Kihun Kim et al. "Seafloor Mapping Using Underwater Video Mosaic". In: *OCEANS 2011* (2011), pp. 8–11.

[25] Litoral Sur and D E L Perú. "Bio-Oceanographic and Fishing Monitoring Made in the Southern Littoral of Peru. MOBOP 0708". In: *Informe Instituto del Mar del Peru (IMARPE)* 39.1-2 (2012), pp. 122–131.

[26] Dorothée Herr, Isensee Kirsten, and Carol Turley. "Ocean Acidification: Overview of the International Policy Landscape and Activities on Ocean Acidification". In: *International Atomic Energy Agency* (June 2013).

[27] Carlos Maldonado. "Water Quality in the Huacho Bay - 2002". In: *Informe Instituto del Mar del Peru (IMARPE)* 39.3-4 (2012), pp. 212–217.

[28] J.D. Jason D Holmes et al. "An Autonomous Underwater Vehicle Towed Array for Ocean Acoustic Measurements and Inversions". In: *Europe Oceans 2005*. Vol. 2. IEEE, 2005, pp. 1058–1061. ISBN: 0-7803-9103-9. DOI: 10.1109/OCEANSE.2005.1513204.

[29] George H Woodman et al. "Acoustic Characteristics of Fish Bombing: Potential to Develop an Automated Blast Detector". In: *Marine Pollution Bulletin* 46.1 (Jan. 2003), pp. 99–106. ISSN: 0025326X. DOI: 10.1016/S0025-326X(02)00322-3.

[30] George H Woodman et al. "A Direction-Sensitive Underwater Blast Detector and Its Application for Managing Blast Fishing." In: *Marine pollution bulletin* 49.11-12 (Dec. 2004), pp. 964–73. ISSN: 0025-326X. DOI: 10.1016/j.marpolbul.2004.06.022. PMID: 15556182.

[31] Francisco Ganoza et al. "Detection and Monitoring of Blast Fishing". In: *Informe Instituto del Mar del Peru (IMARPE)* 42.1 (2015), pp. 74–121.

[32] FAO. *A Fishery Manager's Guidebook*. Second Edition. The Food and Agriculture Organization of the United Nations and Wiley-Blackwell, 2009. 544 pages. ISBN: 978-92-5-105963-0.

[33] Francisco Ganoza et al. "Operation and Performance of Artisanal Purse Seine in Huacho Area". In: *Informe Instituto del Mar del Peru (IMARPE)* 41.1-4 (2014), pp. 82–93.

[34] INEI. “Pesca”. In: *Compendio Estadístico Perú 2014* (2014), pp. 1023–1055.

[35] Carlos Salazar and German Chacon. “Selectivity of Granton Trawl Net 400x120 Mm PA in the Evaluation of Peruvian Hake - Autumn 2002”. In: *Informe Instituto del Mar del Peru (IMARPE)* 39.1-2 (2012), pp. 132–135.

[36] Francisco Ganoza et al. “Monitoreo e Impacto de La Pesca Fantasma En El Litoral Peruano”. In: 41 (2014), pp. 66–75.

[37] Rubén Ercoli et al. “Experiencias de Selectividad En Los Copos de Las Redes de Arrastre y Desarrollo de Dispositivos Selectivos Con Grillas En La Pesqueŕia Argentina”. In: *El Mar Argentino y sus Recursos Pesqueros* 3 (2001), pp. 121–144.

[38] Sara Purca et al. “Relationship between Anchovy and the Environment at Different Temporal Scales”. In: *Boletín Instituto del Mar del Peru (IMARPE)* 25.1-2 (2010).

[39] Violeta León et al. “pH as a Tracer of Biogeochemical Variability in the Humboldt System Violeta”. In: *Boletín Instituto del Mar del Peru (IMARPE)* 26.1-2 (2011), pp. 19–24.

[40] Jorge Quispe and Luis Vásquez. “Índice "LABCOS" Para La Caracterización de Eventos El Niño y La Niña Fretne a La Costa Del Perú, 1976-2015”. In: *Boletín Trimestral Oceanográfico* 1.1-4 (2015), pp. 12–16.

[41] Tony Anculle et al. “Anomaĺias Del Perfil Vertical de Temperatura Del Punto Fijo Paita Como Indicador de La Propagación de Ondas Kelvin”. In: *Boletín Trimestral Oceanográfico* 1.1 - 4 (2015), pp. 6–8.

[42] Guadalupe Sánchez, Violeta Flores, and Áida Henostroza. “Environmental Quality in the Wetland Pool La Arenilla, Callao - 2008”. In: *Informe Instituto del Mar del Peru (IMARPE)* 41.1-4 (2014), pp. 202–214.

[43] Peter M. Chapman. “Future Challenges for Marine Pollution Monitoring and Assessment”. In: *Marine Pollution Bulletin* 95.1 (2015), pp. 1–2. ISSN: 0025326X. DOI: `10.1016/j.marpolbul.2015.05.043`.

[44] Francisco Ganoza et al. “Nivel de Ruido y Efectos En El Ecosistema Por Uso Del Zumbador En La Pesca de Suco Paralonchurus Peruanus, PACASMAYO”. In: Volumen 41.1-4 (2014), pp. 162–178. DOI: `ISSN0378-7702`.

[45] Francisco Ganoza et al. “Gillnets in Coastal Resources with the Use of Buzzers in Pacasmayo”. In: *Informe Instituto del Mar del Peru (IMARPE)* 41.1-4 (2014), pp. 16–23.

[46] Francisco Ganoza et al. “Illegal Fishing in La Libertad and Lambayeque”. In: *Informe Instituto del Mar del Peru (IMARPE)* 41.1-4 (2014).

[47] T.V. Zorikov and N.A. Dubrovsky. "Echo-Processing Procedure in Bottlenose Dolphins". In: *Oceans 2003*. Vol. 1. IEEE, 2003, 320–326 Vol.1. ISBN: 0-933957-30-0. DOI: 10.1109/OCEANS.2003.178577.

[48] Manuel Castellote, Christopher W. Clark, and Marc O. Lammers. "Acoustic and Behavioural Changes by Fin Whales (Balaenoptera Physalus) in Response to Shipping and Airgun Noise". In: *Biological Conservation* 147.1 (Mar. 2012), pp. 115–122. ISSN: 00063207. DOI: 10.1016/j.biocon.2011.12.021.

[49] IMARPE. "Mortandad de Delfines En El Litoral de La Costa Norte, Febrero a Abril Del 2012". In: *Instituto del Mar del Perú* (2012), p. 81.

[50] *Index of /Eyebot7*. URL: http://robotics.ee.uwa.edu.au/eyebot7/ (visited on 02/10/2019).

[51] C. Wang and W. Wang. "An Embedded Controller for a Quadruped Robot Based on ARM and DSP". In: *2016 IEEE International Conference on Mechatronics and Automation*. Aug. 2016, pp. 1090–1095. DOI: 10.1109/ICMA.2016.7558714.

[52] S. Seok et al. "Design Principles for Highly Efficient Quadrupeds and Implementation on the MIT Cheetah Robot". In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 3307–3312. DOI: 10.1109/ICRA.2013.6631038.

[53] Roland Siegwart and Illah R Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Second Edition. Vol. 23. MIT press, 2004. 47–82. ISBN: 0-262-19502-X.

[54] Steven Barrett. *Arduino Microcontroller Processing for Everyone:Part I*. Morgan & Claypool, 2010. 244 pp. ISBN: 978-1-60845-438-9. DOI: 10.2200/S00283ED1V01Y201005DCS029.

[55] Pi-plates. *Pi-Plates Homepage*. 2017. URL: http://pi-plates.com/ (visited on 05/20/2017).

[56] Mikronauts. *Mikronauts.Com - Microcontrollers Explored. RoboPi*. 2017. URL: http://www.mikronauts.com/ (visited on 05/20/2017).

[57] IOIO. *IOIO-OTG Board*. 2017. URL: https://github.com/ytai/ioio/ (visited on 05/22/2017).

[58] Pixhawk. *Pixhawk Px4 Autopilot*. 2017. URL: https://pixhawk.org/ (visited on 05/22/2017).

[59] D. Dobriborsci, A. Kapitonov, and N. Nikolaev. "The Basics of the Identification, Localization and Navigation for Mobile Robots". In: *2017 International Conference on Information and Digital Technologies (IDT)*. July 2017, pp. 100–105. DOI: 10.1109/DT.2017.8024279.

[60] S. Gupta. "Autonomous Path Planning of Differential Drive Robots for Co-Ordinate Based Navigation". In: *2016 IEEE International Conference on Mechatronics and Automation*. Aug. 2016, pp. 563–568. DOI: 10.1109/ICMA.2016.7558625.

[61] J. Heikkinen, T. Minav, and A. D. Stotckaia. "Self-Tuning Parameter Fuzzy PID Controller for Autonomous Differential Drive Mobile Robot". In: *2017 XX IEEE International Conference on Soft Computing and Measurements (SCM)*. May 2017, pp. 382–385. DOI: 10.1109/SCM.2017.7970592.

[62] Y. Ma et al. "Motion Planning for Non-Holonomic Mobile Robots Using the i-PID Controller and Potential Field". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 3618–3623. DOI: 10.1109/IROS.2014.6943069.

[63] Morgan Quigley et al. "ROS: An Open-Source Robot Operating System". In: *ICRA Workshop on Open Source Software*. Vol. 3. Jan. 2009.

[64] D. Ribas et al. "Girona 500 AUV: From Survey to Intervention". In: *IEEE/ASME Transactions on Mechatronics* 17.1 (Feb. 2012), pp. 46–53. ISSN: 1083-4435. DOI: 10.1109/TMECH.2011.2174065.

[65] B. Abhishek et al. "Low Cost ROS Based Semi-Autonomous Drone with Position and Altitude Lock". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI). Sept. 2017, pp. 2109–2112. DOI: 10.1109/ICPCSI.2017.8392087.

[66] Y. Gao et al. "A Patrol Mobile Robot for Power Transformer Substations Based on ROS". In: *2018 Chinese Control And Decision Conference (CCDC)*. 2018 Chinese Control And Decision Conference (CCDC). June 2018, pp. 456–460. DOI: 10.1109/CCDC.2018.8407176.

[67] K. DeMarco, M. E. West, and T. R. Collins. "An Implementation of ROS on the Yellowfin Autonomous Underwater Vehicle (AUV)". In: *OCEANS'11 MTS/IEEE KONA*. OCEANS'11 MTS/IEEE KONA. Sept. 2011, pp. 1–7. DOI: 10.23919/OCEANS.2011.6107001.

[68] N. Koenig and A. Howard. "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. ISSN: Sept. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[69] Carlo Pinciroli et al. "ARGoS: A Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems". In: *Swarm Intelligence* 6.4 (2012), pp. 271–295.

[70] O. Michel. "Webots: Professional Mobile Robot Simulation". In: *International Journal of Advanced Robotic Systems* 1.1 (2004), pp. 39–42.

[71] Mario Prats et al. "An Open Source Tool for Simulation and Supervision of Underwater Intervention Missions". In: *IEEE International Conference on Intelligent Robots and Systems* (2012), pp. 2577–2582. ISSN: 21530858. DOI: 10.1109/IROS.2012.6385788.

[72] T. Braunl et al. "The Autonomous Underwater Vehicle Initiative - Project Mako". In: *IEEE Conference on Robotics, Automation and Mechatronics, 2004.* IEEE Conference on Robotics, Automation and Mechatronics, 2004. Vol. 1. Dec. 2004, 446–451 vol.1. DOI: 10.1109/RAMECH.2004.1438961.

[73] P.G.C. Namal Senarathne et al. "MarineSIM: Robot Simulation for Marine Environments". In: IEEE, May 2010, pp. 1–5. ISBN: 978-1-4244-5221-7. DOI: 10.1109/OCEANSSYD.2010.5603839.

[74] R Smith. *ODE: Open Dynamics Engine.* Jan. 2009. URL: http://www.ode.org/ (visited on 02/06/2018).

[75] Aaron Staranowicz and Gian Luca Mariottini. "A Survey and Comparison of Commercial and Open-Source Robotic Simulator Software". In: ACM Press, 2011, p. 1. ISBN: 978-1-4503-0772-7. DOI: 10.1145/2141622.2141689.

[76] Gregory Junker. *Pro OGRE 3D Programming (Pro).* Berkely, CA, USA: Apress, 2006. ISBN: 1-59059-710-9.

[77] Michael Lewis and Jeffrey Jacobson. "Game Engines in Scientific Research". In: *Communications of the ACM* 45 (Jan. 1, 2002), pp. 27–31.

[78] Chuck Lewin. *Mathematics of Motion Control Profiles.* 2007. URL: www.pmdcorp.com.

[79] Yulin Zhang et al. "Dynamic Model Based Robust Tracking Control of a Differentially Steered Wheeled Mobile Robot". In: *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207).* Vol. 2. June 1998, 850–855 vol.2. DOI: 10.1109/ACC.1998.703528.

[80] Thomas Bräunl. *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 3-540-34318-0.

[81] *Unity.* URL: https://unity3d.com (visited on 07/24/2018).

[82] *Blue Robotics.* URL: https://www.bluerobotics.com/ (visited on 07/30/2018).

[83] Thrun Sebastian et al. "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots". In: *Springer Autonomous Robots* 5.3-4 (1998), pp. 253–271.

[84] H. Durrant-Whyte and T. Bailey. “Simultaneous Localization and Mapping: Part I”. In: *IEEE Robotics Automation Magazine* 13.2 (June 2006), pp. 99–110. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[85] Liam Paull et al. “AUV Navigation and Localization: A Review”. In: *IEEE J. Ocean. Eng.* 39.1 (2014), pp. 131–149. ISSN: 03649059. DOI: 10.1109/JOE.2013.2278891.

[86] Jiang Yan et al. “A Review on Localization and Mapping Algorithm Based on Extended Kalman Filtering”. In: *2009 International Forum on Information Technology and Applications* 2 (May 2009), pp. 435–440. DOI: 10.1109/IFITA.2009.284.

[87] Michael Montemerlo et al. “FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem”. In: *AAAI Innovative Applications of Artificial Intelligence (IAAI)* 593598 (2002).

[88] Fredrik Gustafsson. “Particle Filter Theory and Practice with Positioning Applications”. In: *IEEE Aerospace and Electronic Systems Magazine* 25.7 (July 2010), pp. 53–82. ISSN: 0885-8985. DOI: 10.1109/MAES.2010.5546308.

[89] Giorgio Grisetti et al. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intell. Transp. Syst. Mag.* (2010), pp. 31–43. ISSN: 1939-1390. DOI: 10.1109/MITS.2010.939925.

[90] P. Krishnamurthy and F. Khorrami. “A Self-Aligning Underwater Navigation System Based on Fusion of Multiple Sensors Including DVL and IMU”. In: *2013 9th Asian Control Conference (ASCC)* (June 2013), pp. 1–6. DOI: 10.1109/ASCC.2013.6606318.

[91] YoungJin Heo, Gi-Hyeon Lee, and Jinhyun Kim. “EKF-Based Localization for the Underwater Structure Inspection Robot Using Depth Sensor and IMU”. In: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)* (Urai Nov. 2012), pp. 643–645. DOI: 10.1109/URAI.2012.6463108.

[92] Melike Erol, Luiz Filipe M. Vieira, and Mario Gerla. “AUV-Aided Localization for Underwater Sensor Networks”. In: *International Conference on Wireless Algorithms, Systems and Applications (WASA 2007)* (Aug. 2007), pp. 44–54. DOI: 10.1109/WASA.2007.34.

[93] P.E. E An et al. “New Experimental Results on GPS/INS Navigation for Ocean Voyager II AUV”. In: *Proceedings of Symposium on Autonomous Underwater Vehicle Technology.* IEEE, 1996, pp. 249–255. ISBN: 0-7803-3185-0. DOI: 10.1109/AUV.1996.532422.

[94] Randy Hartman et al. “Tactical Underwater Navigation System (TUNS)”. In: *2008 IEEE/ION Position, Location and Navigation Symposium.* IEEE, 2008, pp. 898–911. ISBN: 978-1-4244-1536-6. DOI: 10.1109/PLANS.2008.4570032.

[95] Andrea Caiti et al. “Localization of Autonomous Underwater Vehicles by Floating Acoustic Buoys: A Set-Membership Approach”. In: *IEEE Journal of Oceanic Engineering* 30.1 (Jan. 2005), pp. 140–152. ISSN: 0364-9059. DOI: 10.1109/JOE.2004.841432.

[96] Hwee-Pink Tan et al. “A Survey of Techniques and Challenges in Underwater Localization”. In: *Ocean Engineering* 38.14 (Oct. 1, 2011), pp. 1663–1676. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2011.07.017.

[97] *Oceanographic Instruments, Hydrographic Equipment, Hydrometric Instrumentation.* URL: https://www.valeport.co.uk/ (visited on 08/10/2018).

[98] Jinjun Rao et al. “Navigation Information Fusion for an AUV in Rivers”. In: *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (Sept. 2012), pp. 83–88. DOI: 10.1109/MFI.2012.6343038.

[99] *Xsens.* URL: https://www.xsens.com/ (visited on 08/09/2018).

[100] Jeff Snyder. “Doppler Velocity Log (DVL) Navigation for Observation-Class ROVs”. In: *OCEANS 2010 MTS/IEEE SEATTLE.* Dvl. IEEE, Sept. 2010, pp. 1–9. ISBN: 978-1-4244-4332-1. DOI: 10.1109/OCEANS.2010.5664561.

[101] Roee Diamant, Lutz Lampe, and Senior Member. “Underwater Localization with Time-Synchronization and Propagation Speed Uncertainties”. In: *IEEE Transactions on Mobile Computing* 12.7 (July 2013), pp. 1257–1269. ISSN: 1536-1233. DOI: 10.1109/TMC.2012.100.

[102] *ROVBUILDER.* URL: http://www.rovbuilder.com/ (visited on 08/09/2018).

[103] Philipp Woock and Christian Frey. “Deep-Sea AUV Navigation Using Side-Scan Sonar Images and SLAM”. In: *OCEANS’10 IEEE SYDNEY.* IEEE, May 2010, pp. 1–8. ISBN: 978-1-4244-5221-7. DOI: 10.1109/OCEANSSYD.2010.5603528.

[104] Angelos Mallios et al. “Pose-Based SLAM with Probabilistic Scan Matching Algorithm Using a Mechanical Scanned Imaging Sonar”. In: *OCEANS 2009-EUROPE.* IEEE, May 2009, pp. 1–6. ISBN: 978-1-4244-2522-8. DOI: 10.1109/OCEANSE.2009.5278219.

[105] David Ribas, Pere Ridao, and José Neira. “Underwater SLAM for Structured Environments Using an Imaging Sonar”. In: *Springer Tracts Adv. Robot.* 1 (2010), pp. 1–5. ISSN: 0717-6163. DOI: 10.1007/s13398-014-0173-7.2. pmid: 15003161.

[106] Nathaniel Fairfield and David Wettergreen. “Active Localization on the Ocean Floor with Multibeam Sonar”. In: *Oceans 2008* (2008), pp. 1–10. DOI: 10.1109/OCEANS.2008.5151853.

[107] David Ribas et al. "SLAM Using an Imaging Sonar for Partially Structured Underwater Environments". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2006, pp. 5040–5045. ISBN: 1-4244-0258-1. DOI: 10.1109/IROS.2006.282532.

[108] David Ribas et al. "Underwater SLAM in a Marina Environment". In: *IEEE Int. Conf. Intell. Robot. Syst.* 2007, pp. 1455–1460. DOI: 10.1109/IROS.2007.4399222.

[109] Hordur Johannsson et al. "Imaging Sonar-Aided Navigation for Autonomous Underwater Harbor Surveillance". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2010, pp. 4396–4403. ISBN: 978-1-4244-6674-0. DOI: 10.1109/IROS.2010.5650831.

[110] Maurice F. Fallon et al. "Relocating Underwater Features Autonomously Using Sonar-Based SLAM". In: *IEEE Journal of Oceanic Engineering* 38.3 (July 2013), pp. 500–513. ISSN: 0364-9059. DOI: 10.1109/JOE.2012.2235664.

[111] I.T. Ruiz et al. "Feature Extraction and Data Association for AUV Concurrent Mapping and Localisation". In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)* 3 (2001), pp. 2785–2790. ISSN: 1050-4729. DOI: 10.1109/ROBOT.2001.933044.

[112] Ioseba Tena Ruiz et al. "Concurrent Mapping and Localization Using Sidescan Sonar". In: *IEEE Journal of Oceanic Engineering* 29.2 (Apr. 2004), pp. 442–456. ISSN: 0364-9059. DOI: 10.1109/JOE.2004.829790.

[113] J.M. Saez et al. "Underwater 3D SLAM through Entropy Minimization". In: *IEEE International Conference on Robotics and Automation*. May. Florida: IEEE, 2006, pp. 3562–3567. ISBN: 0-7803-9505-0. DOI: 10.1109/ROBOT.2006.1642246.

[114] J. Aulinas et al. "Feature Extraction for Underwater Visual SLAM". In: *OCEANS 2011 IEEE - Spain*. OCEANS 2011 IEEE - Spain. June 2011, pp. 1–7. DOI: 10.1109/Oceans-Spain.2011.6003474.

[115] Adrian Bodenmann, Blair Thornton, and Tamaki Ura. "Development of Long Range Color Imaging for Wide Area 3D Reconstructions of the Seafloor". In: *2013 IEEE International Underwater Technology Symposium (UT)* (Mar. 2013), pp. 1–5. DOI: 10.1109/UT.2013.6519824.

[116] Ayoung Kim and Ryan M. Eustice. "Real-Time Visual SLAM for Autonomous Underwater Hull Inspection Using Visual Saliency". In: *IEEE Trans. Robot.* 29.3 (2013), pp. 719–733. ISSN: 15523098. DOI: 10.1109/TRO.2012.2235699.

[117] M. Caccia et al. “Online Video Mosaicing through SLAM for ROVs”. In: *OCEANS 2009-EUROPE*. ISSN: May 2009, pp. 1–6. DOI: 10.1109/OCEANSE.2009.5278217.

[118] Friedrich Fraundorfer et al. “Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications”. In: *IEEE Robotics & Automation Magazine* 19.2 (June 2012), pp. 78–90. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2182810.

[119] Davide Scaramuzza, Friedrich Fraundorfer, and By Davide Scaramuzza. “Visual Odometry [Tutorial]”. In: *IEEE Robotics & Automation Magazine* 18.4 (Dec. 2011), pp. 80–92. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.943233.

[120] Heiko Bulow et al. “Underwater Stereo Data Acquisition and 3D Registration with a Spectral Method”. In: *2013 MTS/IEEE OCEANS - Bergen* (June 2013), pp. 1–7. DOI: 10.1109/OCEANS-Bergen.2013.6608115.

[121] T. Bailey and H. Durrant-Whyte. “Simultaneous Localization and Mapping (SLAM): Part II”. In: *IEEE Robotics Automation Magazine* 13.3 (Sept. 2006), pp. 108–117. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1678144.

[122] B Hiebert-Treuer. “An Introduction to Robot SLAM (Simultaneous Localization And Mapping)”. Bachelor’s Theses. Middlebury, VT, USA: Middlebury College, 2007. 75 pp.

[123] S. Maskell and N. Gordon. “A Tutorial on Particle Filters for On-Line Nonlinear/Non-Gaussian Bayesian Tracking”. In: *IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174)*. Vol. Workshop. ISSN: Oct. 2001, 2/1–2/15 vol.2. DOI: 10.1049/ic:20010246.

[124] M. Montemerlo and Sebastian Thrun. “Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. IEEE, 2003, pp. 1985–1991. ISBN: 0-7803-7736-2. DOI: 10.1109/ROBOT.2003.1241885.

[125] Tim Bailey et al. “Consistency of the EKF-SLAM Algorithm”. In: *IEEE Int. Conf. Intell. Robot. Syst.* 1 (2006), pp. 3562–3568. ISSN: 10504729. DOI: 10.1109/IROS.2006.281644.

[126] Michael Montemerlo et al. “Fast SLAM 2.0 : An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges”. In: *Int. Jt. Conf. Artif. Intell. IJCAI* (2003), pp. 1151–1156.

[127] M.S. Sanjeev Arulampalam et al. “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”. In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188. ISSN: 1053587X. DOI: 10.1109/78.978374.

[128] A Doucet, SJ Godsill, and C Andrieu. *On Sequential Simulation-Based Methods for Bayesian Filtering*. University of Cambridge, Department of Engineering. 1998, pp. 1–26.

[129] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

[130] N. Fairfield, G. Kantor, and D. Wettergreen. "Towards Particle Filter SLAM with Three Dimensional Evidence Grids in a Flooded Subterranean Environment". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (May 2006), pp. 3575–3580. ISSN: 1050-4729. DOI: 10.1109/ROBOT.2006.1642248.

[131] Sebastian Thrun. "Particle Filters in Robotics". In: *In Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*. 2002.

[132] M. Montemerlo, S. Thrun, and W. Whittaker. "Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 1. IEEE, 2002, pp. 695–701. ISBN: 0-7803-7272-7. DOI: 10.1109/ROBOT.2002.1013439.

[133] E. Olson and P. Agarwal. "Inference on Networks of Mixtures for Robust Robot Mapping". In: *The International Journal of Robotics Research* 32.7 (July 2013), pp. 826–840. ISSN: 0278-3649. DOI: 10.1177/0278364913479413.

[134] F. Lu and E. Milios. "Globally Consistent Range Scan Alignment for Environment Mapping". In: *Autonomous Robots* 4.4 (Oct. 1997), pp. 333–349. ISSN: 0929-5593. DOI: 10.1023/A:1008854305733.

[135] Ling Chen, Sen Wang, and Huosheng Hu. "Pose-Based GraphSLAM Algorithm for Robotic Fish with a Mechanical Scanning Sonar". In: *2013 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2013* (December 2013), pp. 38–43. DOI: 10.1109/ROBIO.2013.6739432.

[136] Cyrill Stachniss. *Class Lecture: "Robot Mapping - WS 2013/14" Short Summary.* Freiburg im Breisgau, Germany, 2013.

[137] L. Jaulin. "A Nonlinear Set Membership Approach for the Localization and Map Building of Underwater Robots". In: *IEEE Transactions on Robotics* 25.1 (Feb. 2009), pp. 88–98. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2010358.

[138] Ayoung Kim. "Active Visual SLAM with Exploration for Autonomous Underwater Navigation Real-Time Pose-Graph Visual SLAM". University of Michigan Dept. of Mechanical Engineering, 2012.

[139] Ian Mahon et al. "Efficient View-Based SLAM Using Visual Loop Closures". In: *IEEE Trans. Robot.* 24.5 (2008), pp. 1002–1014. ISSN: 15523098. DOI: 10.1109/TRO.2008.2004888.

[140] R.M. Eustice, O. Pizarro, and H. Singh. "Visually Augmented Navigation for Autonomous Underwater Vehicles". In: *IEEE J. Ocean. Eng.* 33.2 (2008), pp. 103–122. ISSN: 0364-9059. DOI: 10.1109/JOE.2008.923547.

[141] Stephen Barkby et al. "An Efficient Approach to Bathymetric SLAM". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2009), pp. 219–224. DOI: 10.1109/IROS.2009.5354248.

[142] Stephen Barkby et al. "Incorporating Prior Maps with Bathymetric Distributed Particle SLAM for Improved AUV Navigation and Mapping". In: *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology* (2009), pp. 1–7.

[143] Bo He et al. "AUV SLAM and Experiments Using a Mechanical Scanning Forward-Looking Sonar." In: *Sensors (Basel, Switzerland)* 12.7 (Jan. 2012), pp. 9386–410. ISSN: 1424-8220. DOI: 10.3390/s120709386.

[144] T. Bailey. *Matlab Utilities - SLAM Simulations.* URL: http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm (visited on 12/07/2014).

[145] Chedrawi Salim. *Python AI Robots, Graph SLAM.*

[146] A. C. T. Koh et al. "Shallow Waters SLAM Experiments on Meredith AUV Using Forward Looking Sonar". In: *OCEANS 2009.* Oct. 2009, pp. 1–6. DOI: 10.23919/OCEANS.2009.5422422.

[147] Josep Aulinas et al. "Selective Submap Joining for Underwater Large Scale 6-DOF SLAM". In: *Intelligent Robots and Systems* (Oct. 2010), pp. 2552–2557. ISSN: 2153-0858. DOI: 10.1109/IROS.2010.5650438.

[148] M. Walter, F. Hover, and J. Leonard. "SLAM for Ship Hull Inspection Using Exactly Sparse Extended Information Filters". In: *2008 IEEE International Conference on Robotics and Automation.* May 2008, pp. 1463–1470. DOI: 10.1109/ROBOT.2008.4543408.

[149] Antoni Burguera et al. "Underwater SLAM with Robocentric Trajectory Using a Mechanically Scanned Imaging Sonar". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Dvl. IEEE, Sept. 2011, pp. 3577–3582. ISBN: 978-1-61284-456-5. DOI: 10.1109/IROS.2011.6094850.

[150] Bo He et al. "Underwater Simultaneous Localization and Mapping Based on EKF and Point Features". In: *2009 IEEE Int. Conf. Mechatronics Autom. ICMA 2009* (2009), pp. 4845–4850. DOI: 10.1109/ICMA.2009.5246398.

[151] Dariush Forouher et al. "Sonar-Based FastSLAM in an Underwater Environment Using Walls as Features". In: *2011 15th International Conference on Advanced Robotics (ICAR)*. Ieee, June 2011, pp. 588–593. ISBN: 978-1-4577-1159-6. DOI: 10.1109/ICAR.2011.6088563.

[152] Donghwa Lee et al. "Experiments on Localization of an AUV Using Graph-Based SLAM". In: *2013 10th Int. Conf. Ubiquitous Robot. Ambient Intell. URAI 2013* (2013), pp. 526–527. DOI: 10.1109/URAI.2013.6677329.

[153] Gabrielle Inglis et al. "A Pipeline for Structured Light Bathymetric Mapping". In: *Intelligent Robots and Systems (IROS)* (Oct. 2012), pp. 4425–4432. ISSN: 2153-0858. DOI: 10.1109/IROS.2012.6386038.

[154] Angelos Mallios et al. "Probabilistic Sonar Scan Matching SLAM for Underwater Environment". In: *OCEANS'10 IEEE SYDNEY*. IEEE, May 2010, pp. 1–8. ISBN: 978-1-4244-5221-7. DOI: 10.1109/OCEANSSYD.2010.5603650.

[155] K Siantidis. "Side Scan Sonar Based Onboard SLAM System for Autonomous Underwater Vehicles". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Nov. 2016, pp. 195–200. DOI: 10.1109/AUV.2016.7778671.

[156] F Bonin-Font et al. "Stereo SLAM for Robust Dense 3D Reconstruction of Underwater Environments". In: *OCEANS 2015 - Genova*. May 2015, pp. 1–6. DOI: 10.1109/OCEANS-Genova.2015.7271333.

[157] M Meireles et al. "Real Time Visual SLAM for Underwater Robotic Inspection". In: *2014 Oceans - St. John's*. Sept. 2014, pp. 1–5. DOI: 10.1109/OCEANS.2014.7003097.

[158] S. Hong and J. Kim. "Efficient Visual SLAM Using Selective Image Registration for Autonomous Inspection of Underwater Structures". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. ISSN: Nov. 2016, pp. 189–194. DOI: 10.1109/AUV.2016.7778670.

[159] Pep Lluis Negre, Francisco Bonin-Font, and Gabriel Oliver. "Cluster-Based Loop Closing Detection for Underwater Slam in Feature-Poor Regions". In: *Proc. - IEEE Int. Conf. Robot. Autom.* Vol. 2016-June. 2016, pp. 2589–2595. DOI: 10.1109/ICRA.2016.7487416.

[160] J Luo et al. “Data Association for AUV Localization and Map Building”. In: *2010 International Conference on Measuring Technology and Mechatronics Automation*. Vol. 1. Mar. 2010, pp. 886–889. DOI: 10.1109/ICMTMA.2010.300.

[161] Georges Younes et al. “Keyframe-Based Monocular SLAM: Design, Survey, and Future Directions”. In: *Robotics and Autonomous Systems* 98 (Dec. 2017), pp. 67–88. ISSN: 09218890. DOI: 10.1016/j.robot.2017.09.010.

[162] Arturo Gil et al. “A Comparative Evaluation of Interest Point Detectors and Local Descriptors for Visual SLAM”. In: *Machine Vision and Applications* 21.6 (Oct. 2010), pp. 905–920. ISSN: 0932-8092, 1432-1769. DOI: 10.1007/s00138-009-0195-x.

[163] S. A. K. Tareen and Z. Saleem. “A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK”. In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). Mar. 2018, pp. 1–10. DOI: 10.1109/ICOMET.2018.8346440.

[164] Ertuğrul Bayraktar and Pınar Boyraz. “Analysis of Feature Detector and Descriptor Combinations with a Localization Experiment for Various Performance Metrics”. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 25 (2017), pp. 2444–2454. ISSN: 13000632, 13036203. DOI: 10.3906/elk-1602-225.

[165] Alessandro Pieropan et al. “Feature Descriptors for Tracking by Detection: A Benchmark”. In: (July 20, 2016). arXiv: 1607.06178 [cs].

[166] Johan Johansson, Martin Solli, and Atsuto Maki. “An Evaluation of Local Feature Detectors and Descriptors for Infrared Images”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 711–723.

[167] Şahin Işık. “A Comparative Evaluation of Well-Known Feature Detectors and Descriptors”. In: *International Journal of Applied Mathematics, Electronics and Computers* 3.1 (Dec. 18, 2014), p. 1. ISSN: 2147-8228. DOI: 10.18100/ijamec.60004.

[168] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. “Visual Simultaneous Localization and Mapping: A Survey”. In: *Artificial Intelligence Review* 43.1 (Jan. 1, 2015), pp. 55–81. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-012-9365-8.

[169] J. Klippenstein and H. Zhang. “Performance Evaluation of Visual SLAM Using Several Feature Extractors”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct. 2009, pp. 1574–1581. DOI: 10.1109/IROS.2009.5354001.

[170] A Burguera, F Bonin-Font, and G Oliver. "Towards Robust Image Registration for Underwater Visual Slam". In: *Proc. Int. Conf. Comput. Vision, Theory Appl. {(VISAPP)}.* (2014), pp. 539–544. DOI: 10.5220/0004682005390544.

[171] S. M. Luria and Jo Ann S. Kinney. "Underwater Vision". In: *Science* 167.3924 (1970), pp. 1454–1461. ISSN: 0036-8075. JSTOR: 1728965.

[172] Kashif Iqbal et al. "Underwater Image Enhancement Using an Integrated Colour Model." In: *IAENG International Journal of Computer Science* 34.2 (2007).

[173] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Trans. Robot.* 32.6 (2016), pp. 1309–1332. ISSN: 15523098. DOI: 10.1109/TRO.2016.2624754. pmid: 6576973927449638915.

[174] A. Huletski, D. Kartashov, and K. Krinkin. "Evaluation of the Modern Visual SLAM Methods". In: *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT).* Nov. 2015, pp. 19–25. DOI: 10.1109/AINL-ISMW-FRUCT.2015.7382963.

[175] Alberto Quattrini Li et al. "Experimental Comparison of Open Source Vision-Based State Estimation Algorithms". In: *2016 International Symposium on Experimental Robotics.* Ed. by Dana Kulić et al. Vol. 1. Cham: Springer International Publishing, 2017, pp. 775–786. ISBN: 978-3-319-50114-7 978-3-319-50115-4. DOI: 10.1007/978-3-319-50115-4_67.

[176] K. Mikolajczyk and C. Schmid. "A Performance Evaluation of Local Descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (Oct. 2005), pp. 1615–1630. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.188.

[177] Yan Ke and R. Sukthankar. "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. Vol. 2. June 2004, II–506–II–513 Vol.2. DOI: 10.1109/CVPR.2004.1315206.

[178] Xiaozhi Qu et al. "Evaluation of SIFT and SURF for Vision Based Localization". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B3 (June 10, 2016), pp. 685–692. ISSN: 2194-9034. DOI: 10.5194/isprsarchives-XLI-B3-685-2016.

[179] D. G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision.* Proceedings of the Seventh IEEE International Conference on Computer Vision. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[180] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[181] Ethan Rublee et al. "ORB: An Efficient Alternative to SIFT or SURF". In: *Proc. IEEE Int. Conf. Comput. Vis.* 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544. pmid: 20033598.

[182] Pablo F Alcantarilla and T Solutions. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34.7 (2011), pp. 1281–1298.

[183] S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary Robust Invariant Scalable Keypoints". In: *2011 International Conference on Computer Vision*. 2011 International Conference on Computer Vision. Nov. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[184] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill. "SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset". In: IEEE, Dec. 2011, pp. 501–506. ISBN: 978-1-4577-2006-2 978-0-7695-4588-2. DOI: 10.1109/DICTA.2011.90.

[185] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94.

[186] P. Viola and M. Jones. "Rapid Object Detection Using a Boosted Cascade of Simple Features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, I–511–I–518 vol.1. DOI: 10.1109/CVPR.2001.990517.

[187] Shatadal Ghosh et al. "Reliable Pose Estimation of Underwater Dock Using Single Camera: A Scene Invariant Approach". In: *Machine Vision and Applications* 27.2 (Feb. 2016), pp. 221–236. ISSN: 1432-1769. DOI: 10.1007/s00138-015-0736-4.

[188] Alberto Ortiz, Miquel Simó, and Gabriel Oliver. "A Vision System for an Underwater Cable Tracker". In: *Machine Vision and Applications* 13.3 (July 2002), pp. 129–140. ISSN: 1432-1769. DOI: 10.1007/s001380100065.

[189] E. Trabes and M. A. Jordan. “Self-Tuning of a Sunlight-Deflickering Filter for Moving Scenes Underwater”. In: *2015 XVI Workshop on Information Processing and Control (RPIC)*. 2015 XVI Workshop on Information Processing and Control (RPIC). Oct. 2015, pp. 1–6. DOI: 10.1109/RPIC.2015.7497107.

[190] Chourmouzios Tsiotsios et al. “Effective Backscatter Approximation for Photometry in Murky Water”. In: (Apr. 29, 2016). arXiv: 1604.08789 [cs].

[191] Ruoqian Wang et al. “Review on Underwater Image Restoration and Enhancement Algorithms”. In: *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*. ICIMCS ’15. New York, NY, USA: ACM, 2015, 56:1–56:6. ISBN: 978-1-4503-3528-7. DOI: 10.1145/2808492.2808548.

[192] N. Gracias et al. “A Motion Compensated Filtering Approach to Remove Sunlight Flicker in Shallow Water Images”. In: *OCEANS 2008*. OCEANS 2008. Sept. 2008, pp. 1–7. DOI: 10.1109/OCEANS.2008.5152111.

[193] Zhenyou Dai, Xin Wang, and Jian Yang. “Approach to Sunflicker Removal for Underwater Image”. In: *Journal of Electronic Imaging* 24.6 (Nov. 2015), p. 061206. ISSN: 1017-9909, 1560-229X. DOI: 10.1117/1.JEI.24.6.061206.

[194] A Shihavuddin, Nuno Gracias, and Rafael Garcia. “Online Sunflicker Removal Using Dynamic Texture Prediction”. In: *VISAPP 2012 - Proceedings of the International Conference on Computer Vision Theory and Applications* 1 (Jan. 2012).

[195] Hao Zhang. “Removing Backscatter to Enhance the Visibility of Underwater Object”. Thesis. Nanyang Technological University, Aug. 16, 2016.

[196] Chen Qu et al. “Robust Dehaze Algorithm for Degraded Image of CMOS Image Sensors”. In: *Sensors* 17.10 (Sept. 22, 2017), p. 2175. DOI: 10.3390/s17102175.

[197] R. He et al. “Single Image Dehazing with White Balance Correction and Image Decomposition”. In: *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*. ISSN: Dec. 2012, pp. 1–7. DOI: 10.1109/DICTA.2012.6411690.

[198] Jin-Hwan Kim et al. “Optimized Contrast Enhancement for Real-Time Image and Video Dehazing”. In: *Journal of Visual Communication and Image Representation* 24.3 (Apr. 2013), pp. 410–425. ISSN: 10473203. DOI: 10.1016/j.jvcir.2013.02.004.

[199] X. Ding et al. “Underwater Image Dehaze Using Scene Depth Estimation with Adaptive Color Correction”. In: *OCEANS 2017 - Aberdeen*. OCEANS 2017 - Aberdeen. June 2017, pp. 1–5. DOI: 10.1109/OCEANSE.2017.8084665.

[200] T. Treibitz and Y. Y. Schechner. “Active Polarization Descattering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.3 (Mar. 2009), pp. 385–399. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.85.

[201] T. Treibitz and Y. Y. Schechner. “Instant 3Descatter”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06). Vol. 2. 2006, pp. 1861–1868. DOI: 10.1109/CVPR.2006.155.

[202] Z. Murez et al. “Photometric Stereo in a Scattering Medium”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.9 (Sept. 2017), pp. 1880–1891. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2613862.

[203] C. Ancuti et al. “Enhancing Underwater Images and Videos by Fusion”. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, June 2012, pp. 81–88. ISBN: 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1. DOI: 10.1109/CVPR.2012.6247661.

[204] S. Se, D. Lowe, and J. Little. “Vision-Based Mobile Robot Localization and Mapping Using Scale-Invariant Features”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). Vol. 2. 2001, 2051–2058 vol.2. DOI: 10.1109/ROBOT.2001.932909.

[205] J. V. Miro, W. Zhou, and G. Dissanayake. “Towards Vision Based Navigation in Large Indoor Environments”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct. 2006, pp. 2096–2102. DOI: 10.1109/IROS.2006.282487.

[206] G. Bradski. *The OpenCV Library*. 2000. URL: http://www.drdobbs.com/open-source/the-opencv-library/184404319 (visited on 11/07/2018).

[207] Labbé, M. *Find-Object*. accessed 2018-02-10. 2011. URL: http://introlab.github.io/find-object (visited on 02/10/2018).

[208] J. Martínez-Carranza et al. “Towards Autonomous Flight of Micro Aerial Vehicles Using ORB-SLAM”. In: *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. ISSN: Nov. 2015, pp. 241–248. DOI: 10.1109/RED-UAS.2015.7441013.

[209] X. Zuo et al. “Robust Visual SLAM with Point and Line Features”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 1775–1782. DOI: 10.1109/IROS.2017.8205991.

[210] G. x Xin et al. “A RGBD SLAM Algorithm Combining ORB with PROSAC for Indoor Mobile Robot”. In: *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*. Vol. 01. Dec. 2015, pp. 71–74. DOI: 10.1109/ICCSNT.2015.7490710.

[211] Mark Maimone, Yang Cheng, and Larry Matthies. “Visual Odometry on the Mars Exploration Rovers”. In: *2005 IEEE Int. Conf. Syst. Man Cybern.* 1 (2007), pp. 903–910. ISSN: 1070-9932. DOI: 10.1002/rob.20184.

[212] Felipe Guth et al. “Underwater SLAM: Challenges, State of the Art, Algorithms and a New Biologically-Inspired Approach”. In: *5th IEEE RAS/EMBS Int. Conf. Biomed. Robot. Biomechatronics.* 2014, pp. 981–986. DOI: 10.1109/BIOROB.2014.6913908.

[213] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. “DTAM: Dense Tracking and Mapping in Real-Time”. In: *Proc. IEEE Int. Conf. Comput. Vis.* (2011), pp. 2320–2327. ISSN: 1550-5499. DOI: 10.1109/ICCV.2011.6126513. pmid: 6126513.

[214] M. J. Milford, G. F. Wyeth, and D. Prasser. “RatSLAM: A Hippocampal Model for Simultaneous Localization and Mapping”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004.* Vol. 1. Apr. 2004, 403–408 Vol.1. DOI: 10.1109/ROBOT.2004.1307183.

[215] Jakob Engel and Daniel Cremers. “Large-Scale Direct SLAM with Stereo Cameras”. In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst.* Hamburg, 2015, pp. 1935–1942. ISBN: 978-1-4799-9994-1.

[216] Raul Mur-Artal, J. M M Montiel, and Juan D. Tardos. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Trans. Robot.* 31.5 (2015), pp. 1147–1163. ISSN: 15523098. DOI: 10.1109/TRO.2015.2463671. pmid: 21736739.

[217] Georg Klein and David Murray. “Parallel Tracking and Mapping for Small AR Workspaces”. In: *2007 6th IEEE ACM Int. Symp. Mix. Augment. Reality, ISMAR.* 2007. DOI: 10.1109/ISMAR.2007.4538852.

[218] Alejo Concha and Javier Civera. “DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence”. In: *IEEE Int. Conf. Intell. Robot. Syst.* 2015-Decem (June 2015), pp. 5686–5693. ISSN: 21530866. DOI: 10.1109/IROS.2015.7354184.

[219] Stephen M. Chaves et al. “Opportunistic Sampling-Based Active Visual SLAM for Underwater Inspection”. In: *Auton. Robots* 40.7 (2016), pp. 1245–1265. ISSN: 15737527. DOI: 10.1007/s10514-016-9597-6.

[220] Luan Silveira et al. "An Open-Source Bio-Inspired Solution to Underwater SLAM". In: *IFAC-PapersOnLine* 28.2 (2015), pp. 212–217. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.06.035.

[221] Rafael Garcia and Nuno Gracias. "Detection of Interest Points in Turbid Underwater Images". In: *OCEANS* (2011). DOI: 10.1109/Oceans-Spain.2011.6003605.

[222] Rainer Kümmerle et al. "G2o: A General Framework for Graph Optimization". In: *Proc. - IEEE Int. Conf. Robot. Autom.* (June 2011), pp. 3607–3613. ISSN: 10504729. DOI: 10.1109/ICRA.2011.5979949.

[223] A. Geiger et al. "Vision Meets Robotics: The KITTI Dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237. DOI: 10.1177/0278364913491297.

[224] Joel Reis et al. "Design and Experimental Validation of a USBL Underwater Acoustic Positioning System". In: *Sensors* 16.9-1491 (2016). ISSN: 1424-8220. DOI: 10.3390/s16091491.

[225] Raul Mur-Artal and Juan D. Tardos. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *Proc. IEEE Int. Conf. Comput. Vis.* (2016). ISSN: 15523098. DOI: 10.1109/TRO.2012.2197158. pmid: 309728700020.

[226] Wei Tan. "Robust Monocular SLAM in Dynamic Environments". In: *IEEE Int. Symp. Mix. Augment. Real. 2013.* 2013. DOI: 10.1109/ISMAR.2013.6671781.

[227] Western Australian Museum. *Shipwreck Databases.* 2017. URL: http://www.museum.wa.gov.au/maritime-archaeology-db/wrecks/omeo (visited on 10/19/2017).

[228] Huimin Lu et al. "Underwater Optical Image Processing: A Comprehensive Review". In: *Mobile Networks and Applications* 22.6 (Dec. 2017), pp. 1204–1211. ISSN: 1572-8153. DOI: 10.1007/s11036-017-0863-4.

[229] Huimin Lu et al. "Contrast Enhancement for Images in Turbid Water". In: *J. Opt. Soc. Am. A* 32.5 (May 2015), pp. 886–893. DOI: 10.1364/JOSAA.32.000886.

[230] Jeonghwe Gu, Hangil Joe, and Son-Cheol Yu. "On Preprocessing Methods for Feature-Based Photo-Mosaic Underwater". In: *OCEANS 2016 MTS/IEEE Monterey.* Sept. 2016, pp. 1–4. DOI: 10.1109/OCEANS.2016.7761173.

[231] Y. Li and S. Wang. "Underwater Polarization Imaging Technology". In: *2009 Conference on Lasers Electro Optics The Pacific Rim Conference on Lasers and Electro-Optics.* ISSN: Aug. 2009, pp. 1–2. DOI: 10.1109/CLEOPR.2009.5292718.

[232] Seonwook Park, Thomas Schöps, and Marc Pollefeys. “Illumination Change Robustness in Direct Visual SLAM”. In: *ICRA(International Conf. Robot. Autom.* Singapore, 2017. DOI: 10.1109/ICRA.2017.7989525.

[233] Colin McManus et al. “Shady Dealings: Robust, Long-Term Visual Localisation Using Illumination Invariance”. In: *Proc. - IEEE Int. Conf. Robot. Autom.* (2014), pp. 901–906. ISSN: 10504729. DOI: 10.1109/ICRA.2014.6906961.

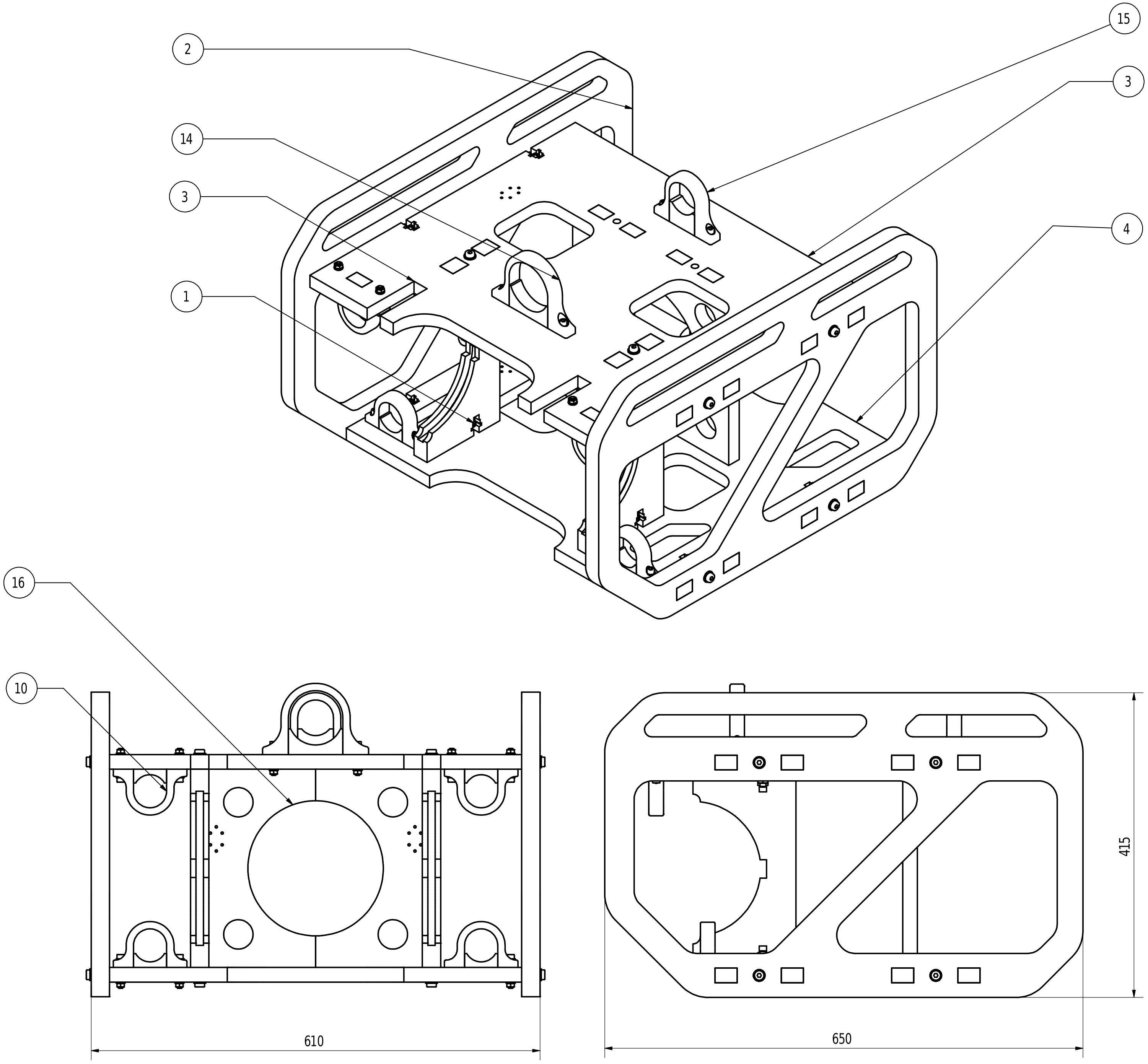
[234] Sivalogeswaran Ratnasingam and T. Martin McGinnity. “Chromaticity Space for Illuminant Invariant Recognition”. In: *IEEE Trans. Image Process.* 21.8 (2012), pp. 3612–3623. ISSN: 10577149. DOI: 10.1109/TIP.2012.2193135. pmid: 22481826.

[235] Austin Eliazar and Ronald Parr. “DP-SLAM 2.0”. In: *IEEE Int. Conf. Robot. Autom.* 2 (2004), pp. 1314–1320. ISSN: 1050-4729. DOI: 10.1109/ROBOT.2004.1308006.

[236] Raul Mur-Artal and Juan D. Tardos. “Visual-Inertial Monocular SLAM with Map Reuse”. In: *IEEE Robot. Autom. Lett.* (2017). ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2653359.

Appendix A

ROV Mechanical Drawings and Connection Diagram

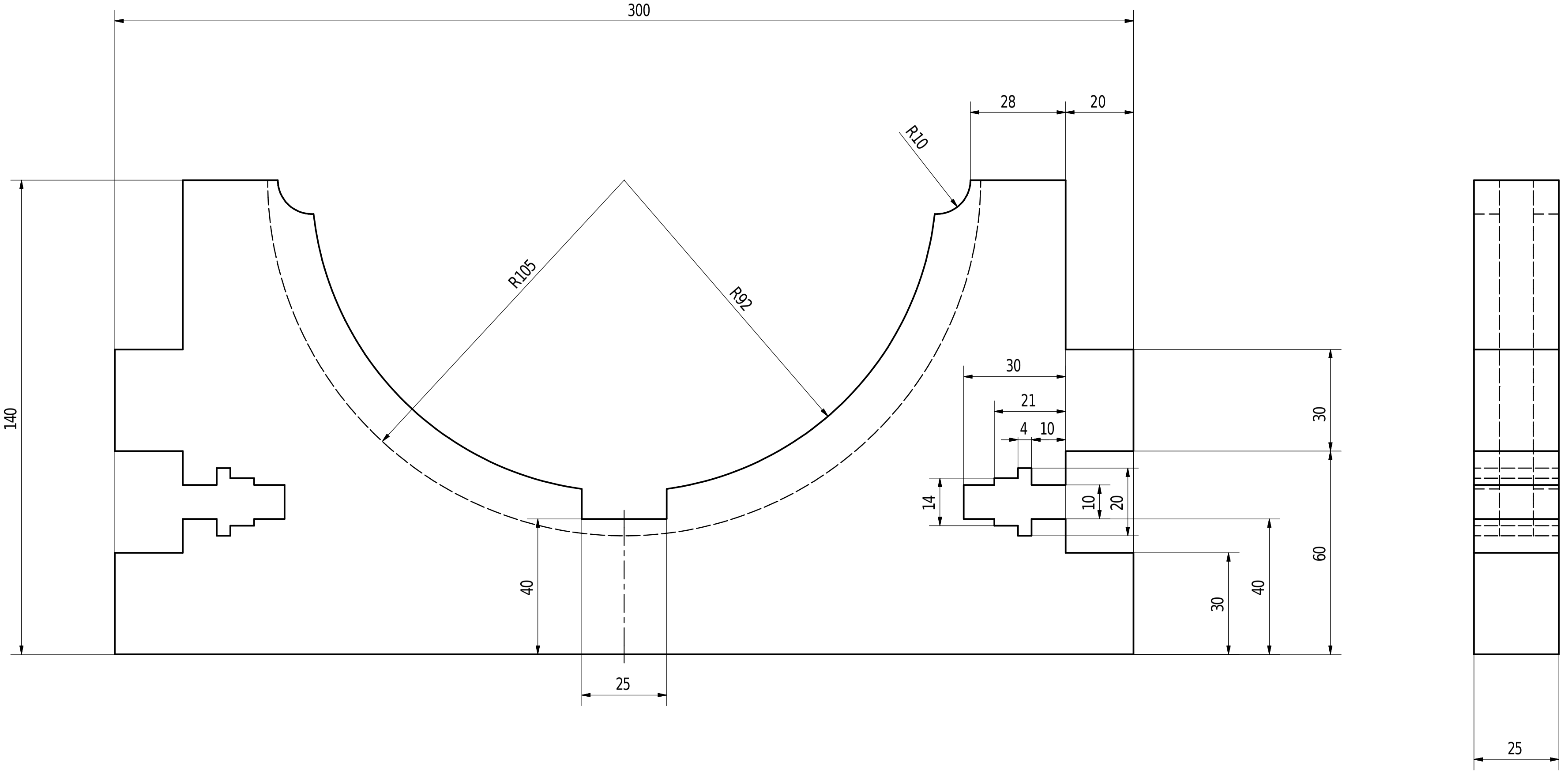


PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	2	Compartment_Side_	
2	2	Side_ROV	
3	1	Up_Main	
4	1	Down_Main	
6	24	ISO 7089 - 8 - 140 HV	Plain washers - Normal series - Product grade A
7	12	ISO 4032 - M8	Hexagon nuts, style 1 - Product grades A and B
8	8	DIN 7984 - M8 x 50	Cylinder Head Cap Screw
9	4	DIN 7984 - M8 x 45	Cylinder Head Cap Screw
10	4	R_Brack_Led	
11	22	ISO 7089 - 6 - 140 HV	Plain washers - Normal series - Product grade A
12	11	ISO 4762 - M6 x 40	Hexagon Socket Head Cap Screw
13	11	ISO 4032 - M6	Hexagon nuts, style 1 - Product grades A and B
14	1	R_Brack_Eureka	
15	1	R_Brack_Eureka_2	
16	2	Middle	

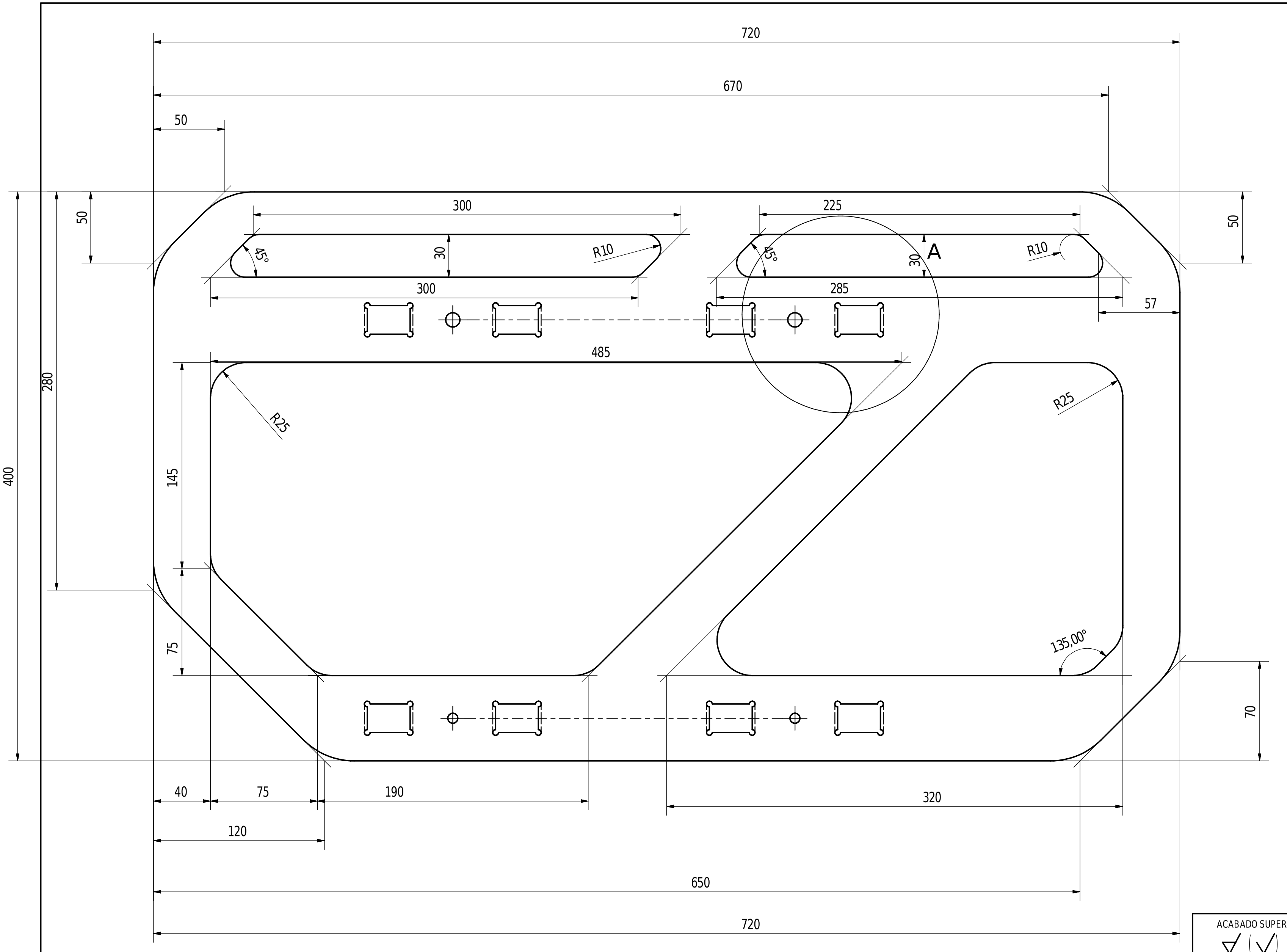
ACABADO SUPERFICIAL ▽	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL HDPE

PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA

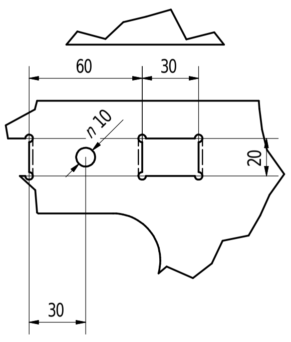
METODO DE PROYECCION	FINCyT N°203 Invetigación Aplicada ASSEMBLY	ESCALA: 1:1
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L10-A2



ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL AISI 304
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA		
METODO DE PROYECCION	FINCYT N°206 INVESTIGACIÓN APLICADA 01 - Comparment Side	ESCALA: (1:3)
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L01-A3



A (1 : 2)

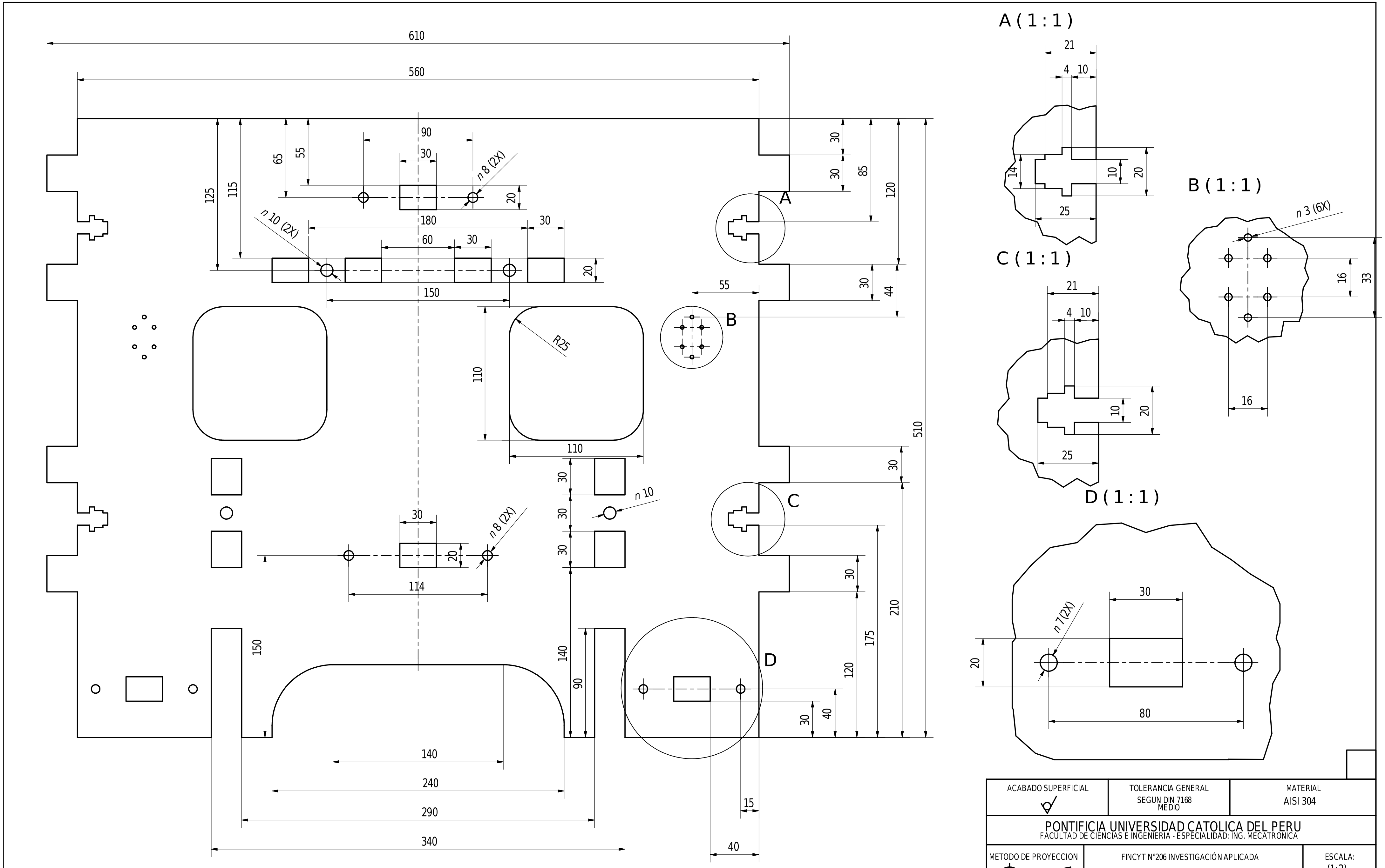


Width 30mm

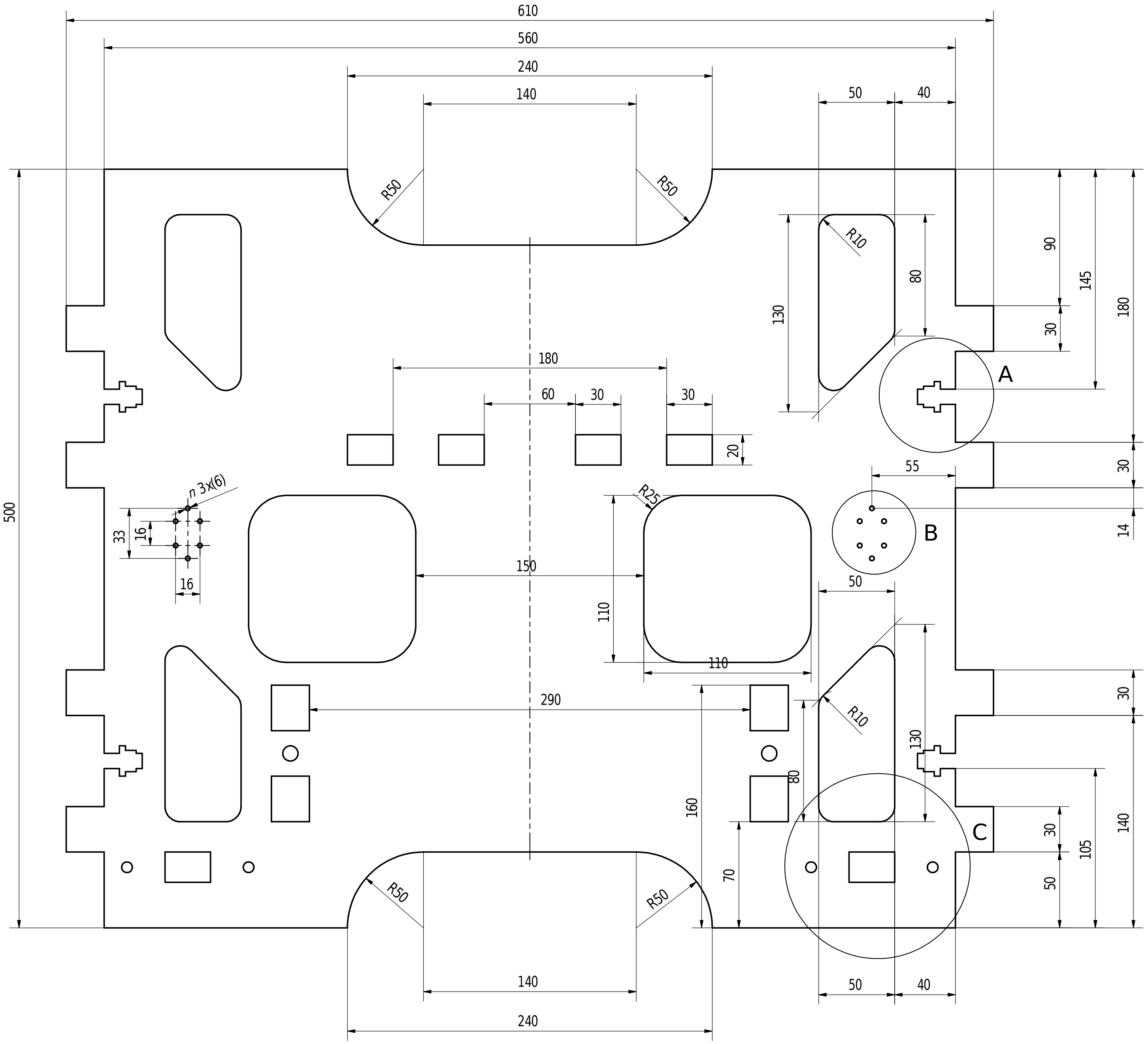
ACABADO SUPERFICIAL ∇ (√)	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL HDPE

PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA

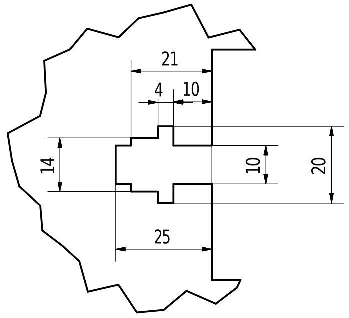
METODO DE PROYECCION	FINCYT N°206 INVESTIGACIÓN APLICADA 02 - Side ROV	ESCALA: (1:3)
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L01-A3



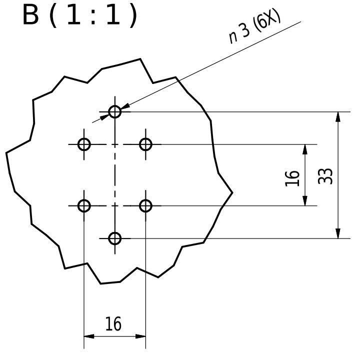
ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL AISI 304
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA		
METODO DE PROYECCION	FINCYT N°206 INVESTIGACIÓN APLICADA 03 - Up Main	ESCALA: (1:3)
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L01-A3



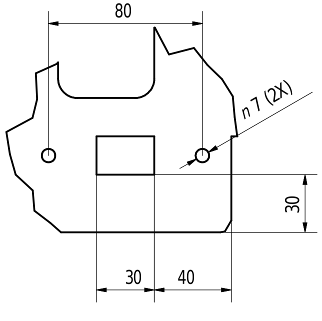
A (1 : 1)



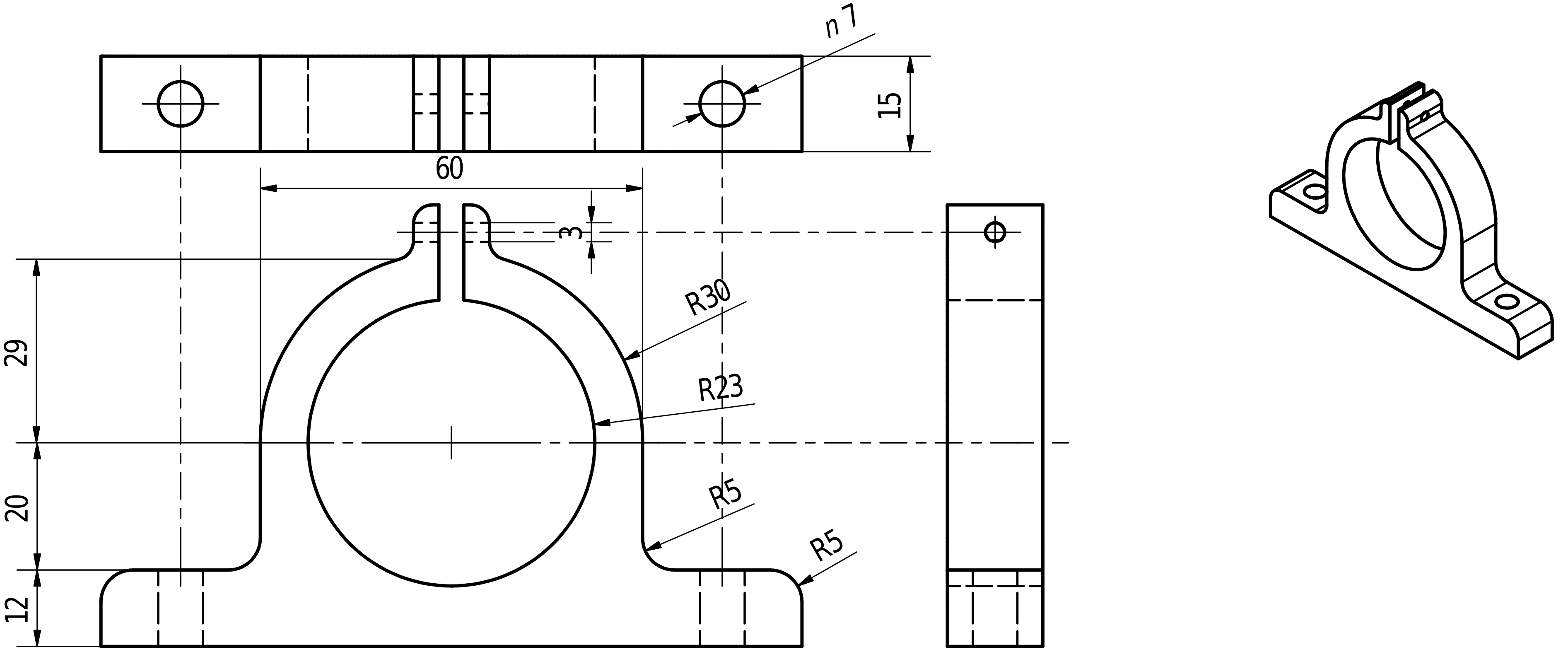
B (1 : 1)



C (1 : 2)



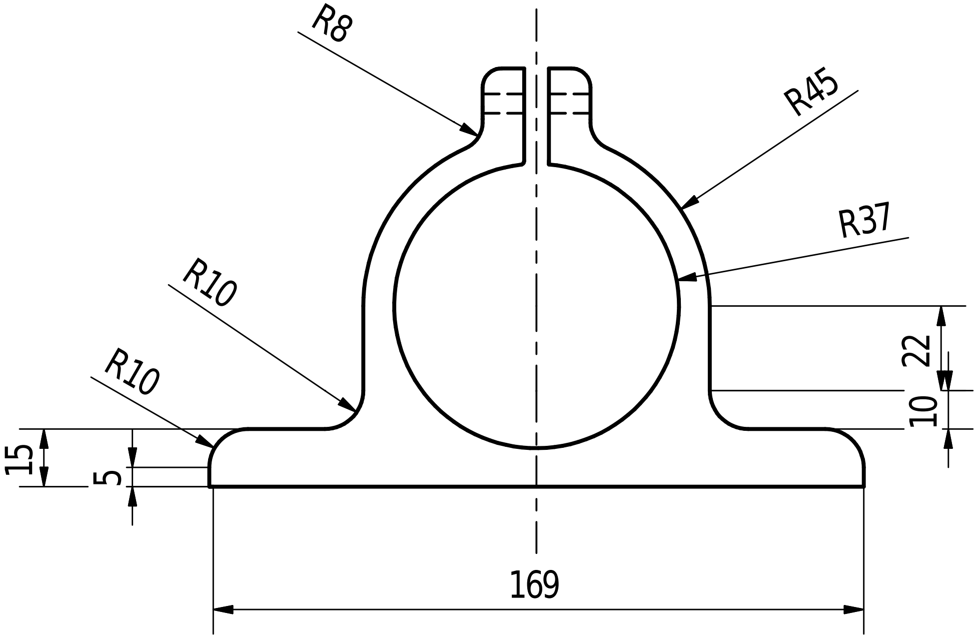
ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL AISI 304
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA		
METODO DE PROYECCION	FINCYT N°206 INVESTIGACIÓN APLICADA 04 - Down main	ESCALA: (1:3)
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L01-A3

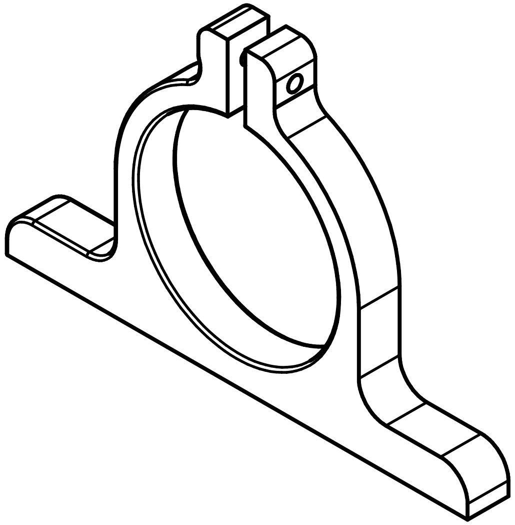


ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL HDP

PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA

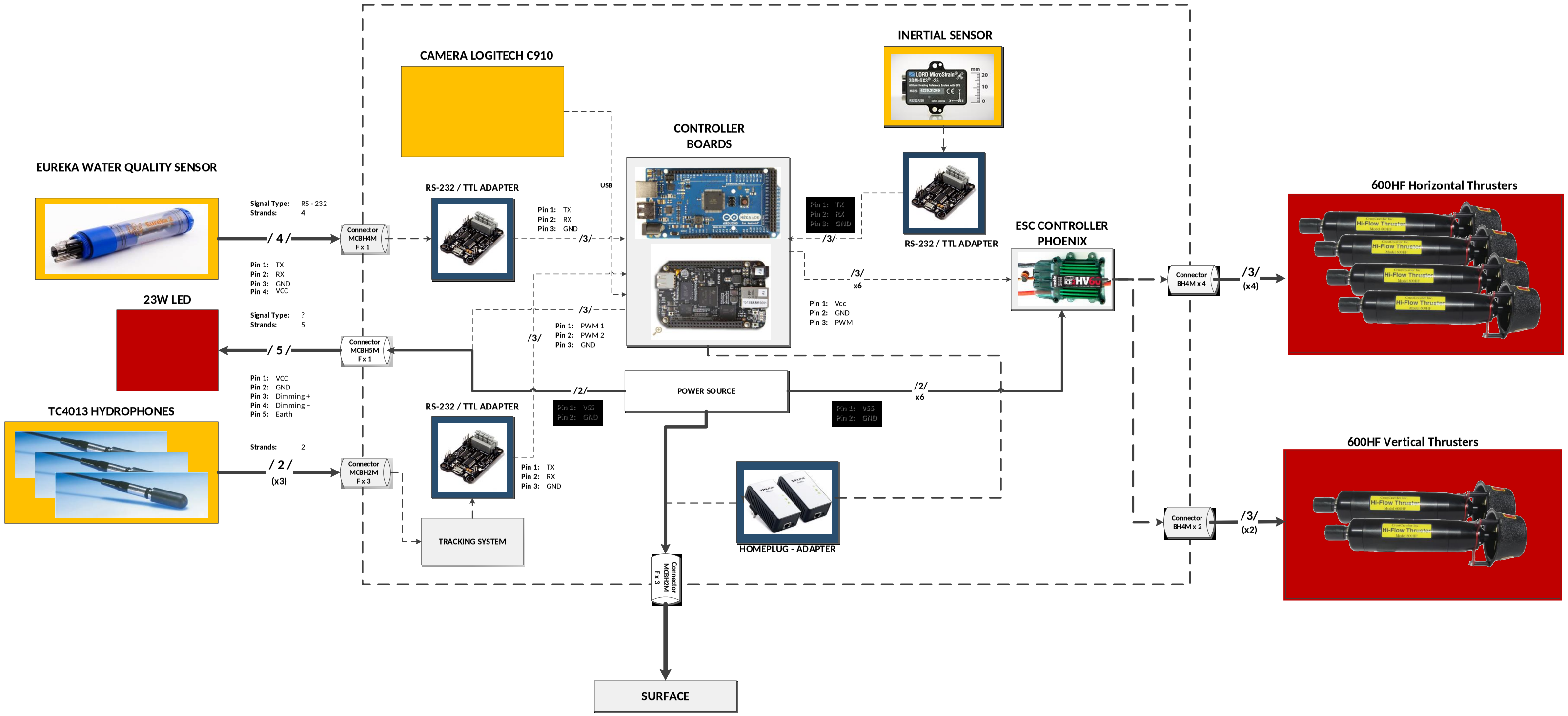
METODO DE PROYECCION	FINCyT N°206 Investigación Aplicada 10 Brack LED	ESCALA: 1:1
DISEÑADO	ARROYO LOPEZ, DANTE A.	FECHA: 2015.02.08
APROBADO	FH / FC	LAMINA: L08-A4





ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGUN DIN 7168 MEDIO	MATERIAL HDP	
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU FACULTAD DE CIENCIAS E INGENIERIA - ESPECIALIDAD: ING. MECATRONICA			
METODO DE PROYECCION	FINCyT N°206 Investigación Aplicada 14 - Brack EUREKA		ESCALA: 1:1
DISEÑADO	ARROYO LOPEZ, DANTE A.		FECHA: 2015.02.08
APROBADO	FH / FC		LAMINA: L08-A4

Connection Diagram



Appendix B

ORB-SLAM2 Experiments: Bar Graph Evaluation

Appendix B

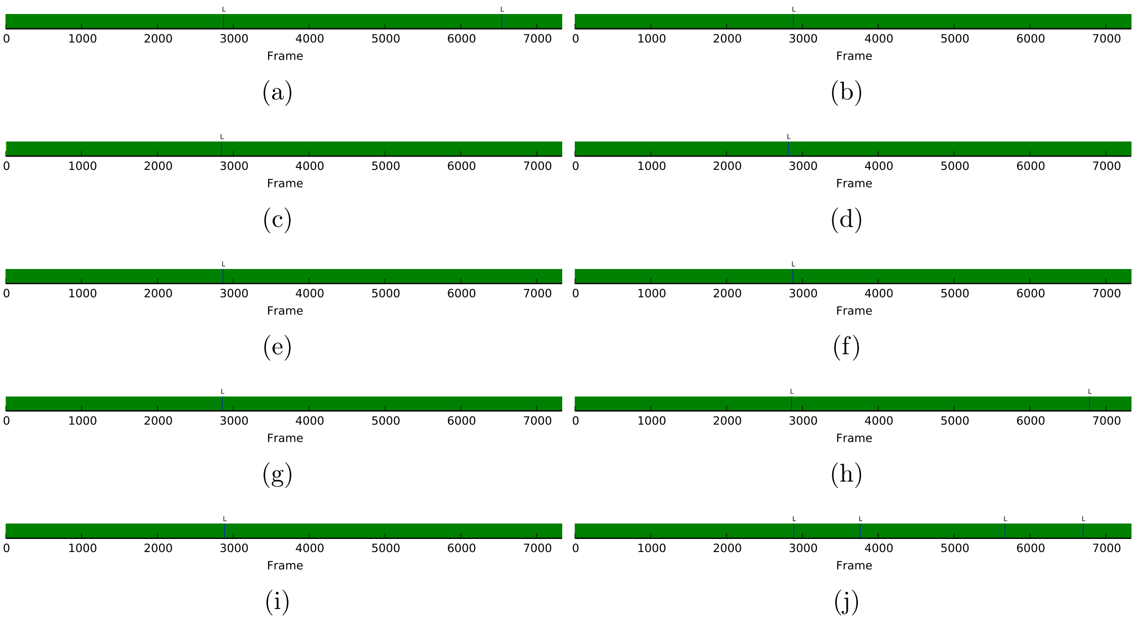
The following Appendix presents the evaluation of the bar graphs of ORB-SLAM2 applied 10 times to different 12 underwater images datasets (Fig. 1 to Fig. 12) to support the results presented in Chapter 6. Please refer to Section 6.4.2 for a detailed explanation of the tests and the definition of the bar graph.

From Fig. 1 to Fig. 4 the bar graphs present the same behavior. The initialization part (in yellow) requires special conditions such as higher number of matches and inliers to compute the Homography or Fundamental Matrix that will rule the tracking process afterwards. Therefore, it is possible to find different performances at this stage as shown in Fig. 4(a) and (h) where the difference is up to 50 frames to initialize.

From Fig. 5 to Fig. 8 the behaviors are the same but, differ in a larger number of frames such as the case of Fig. 4(f) which takes 2000 frames to relocalize compared to the average (less than 50 frames) to relocalize for the rest of tests.

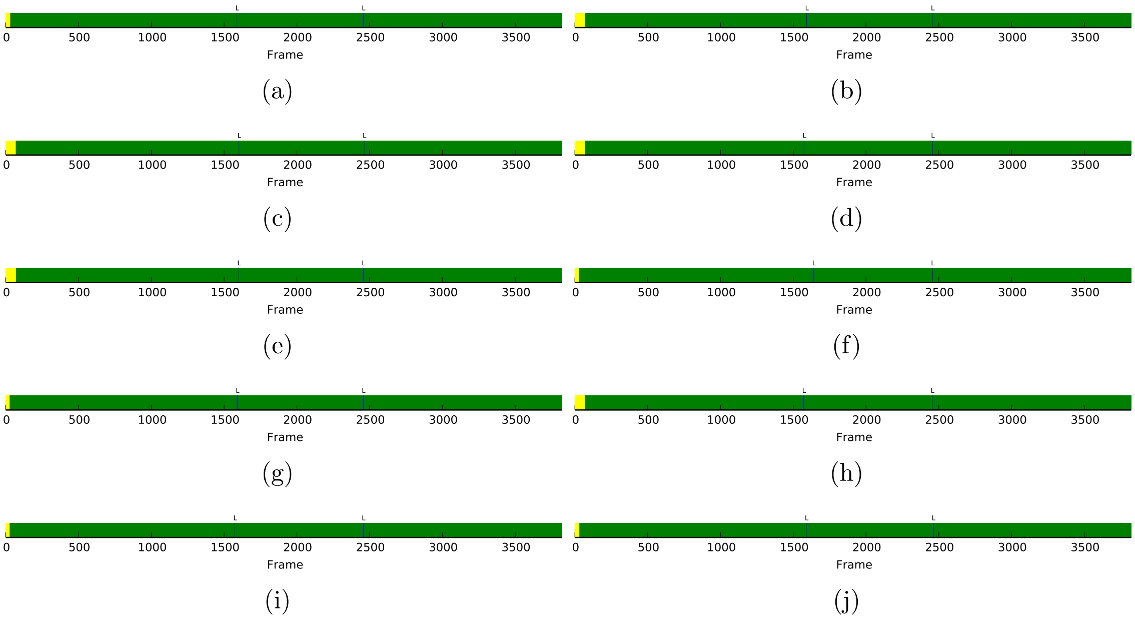
From Fig. 9 to Fig. 12 a variety of behaviors is observed but, they all present problems of losing track at certain point. Fig. 9(e) shows around 700 frames of lost track compared to no lose of track for other 8 runs.

In conclusion, the application of RANSAC at the time of defining the number of inliers in connected frames makes ORB-SLAM2 a highly non-deterministic algorithm as proven experimentally with the previous results. Nevertheless, the largest variations occur when the images are highly affected by the challenges of dealing with underwater images. Therefore, under favourable conditions of the underwater images, such as uniform illumination and low monotony, the algorithm will provide a consistent result despite its non-deterministic nature. Additionally, if the algorithm presents several losses of track and the need of relocalization, it is very likely to be a bad dataset for applying ORB-SLAM2 due to the challenges described in Chapter 5 and 6.



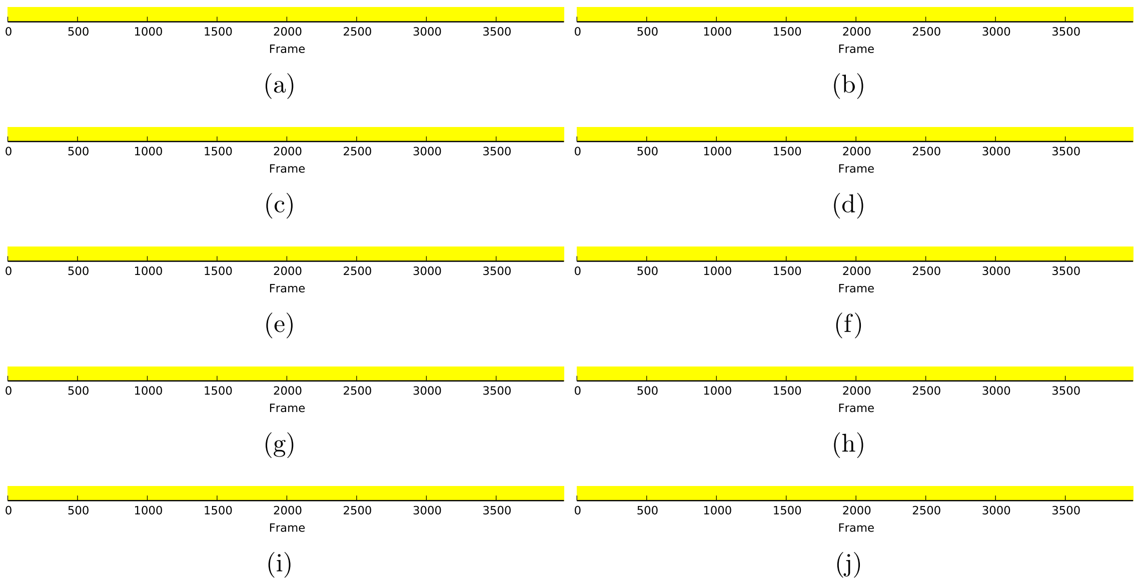
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 1: Dataset 1



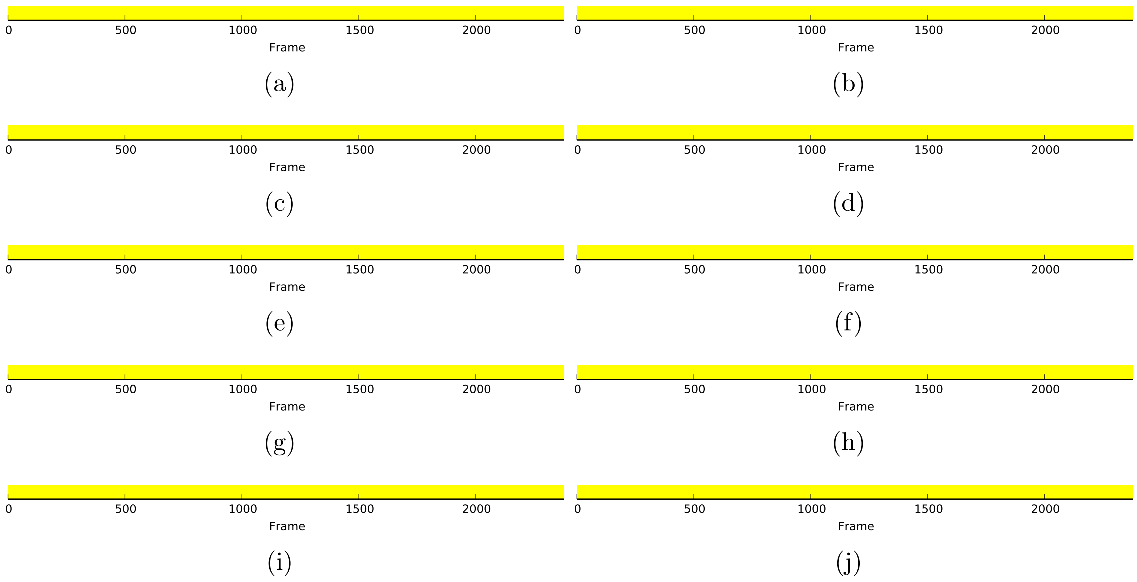
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 2: Dataset 2



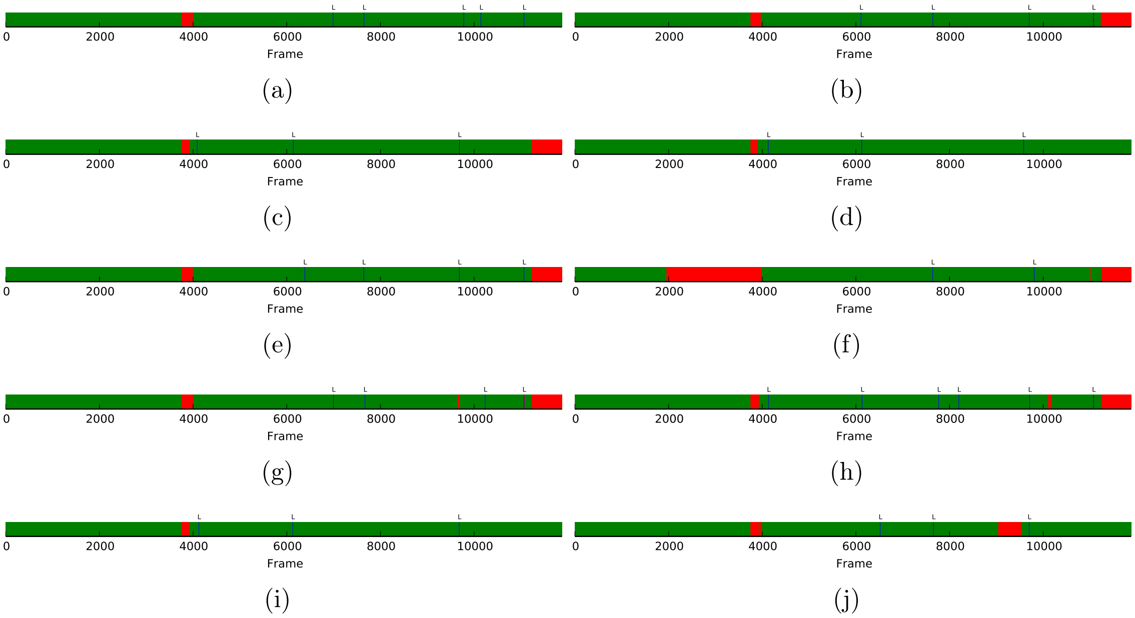
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 3: Dataset 3



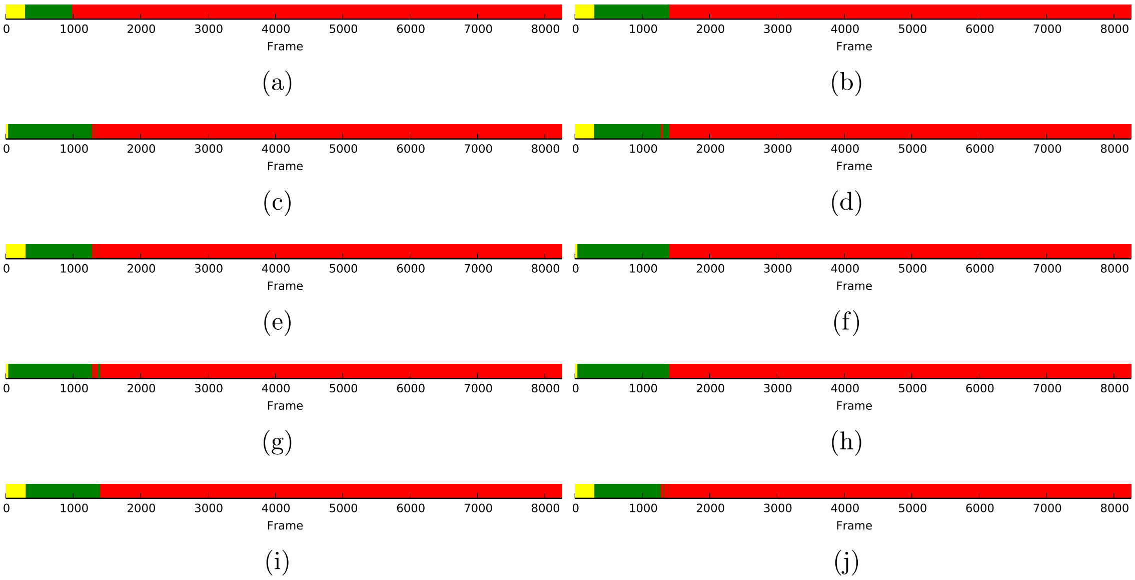
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4: Dataset 4



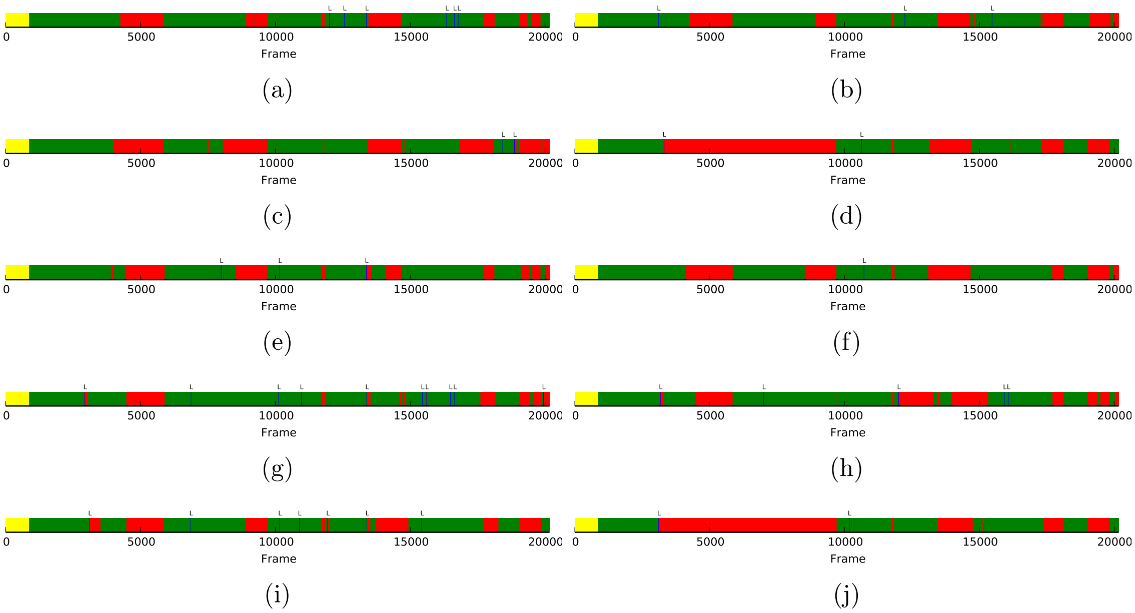
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 5: Dataset 5



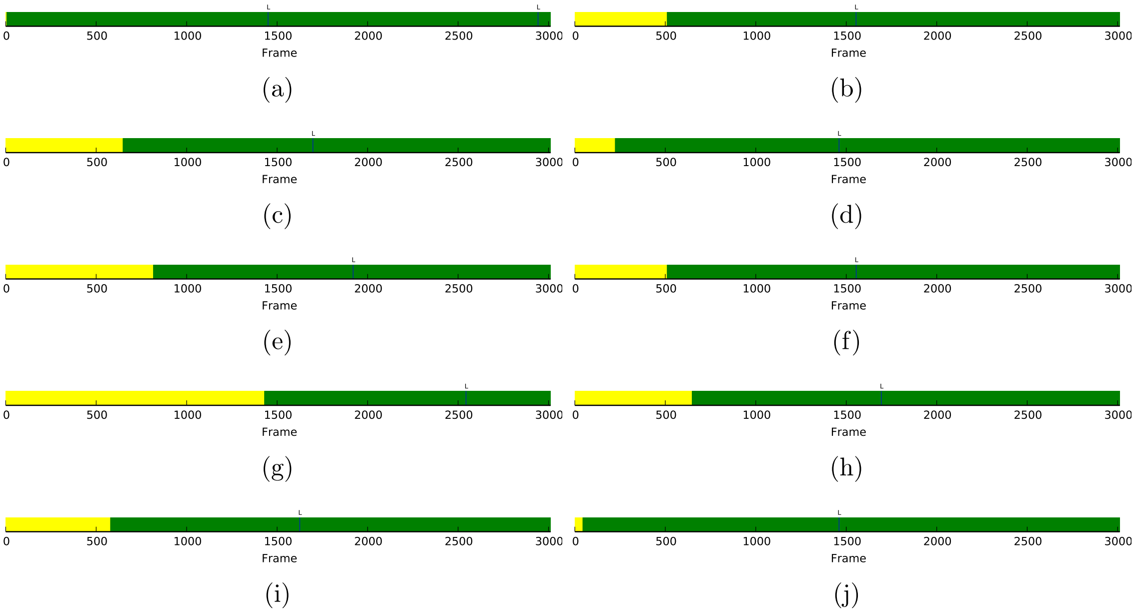
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 6: Dataset 6



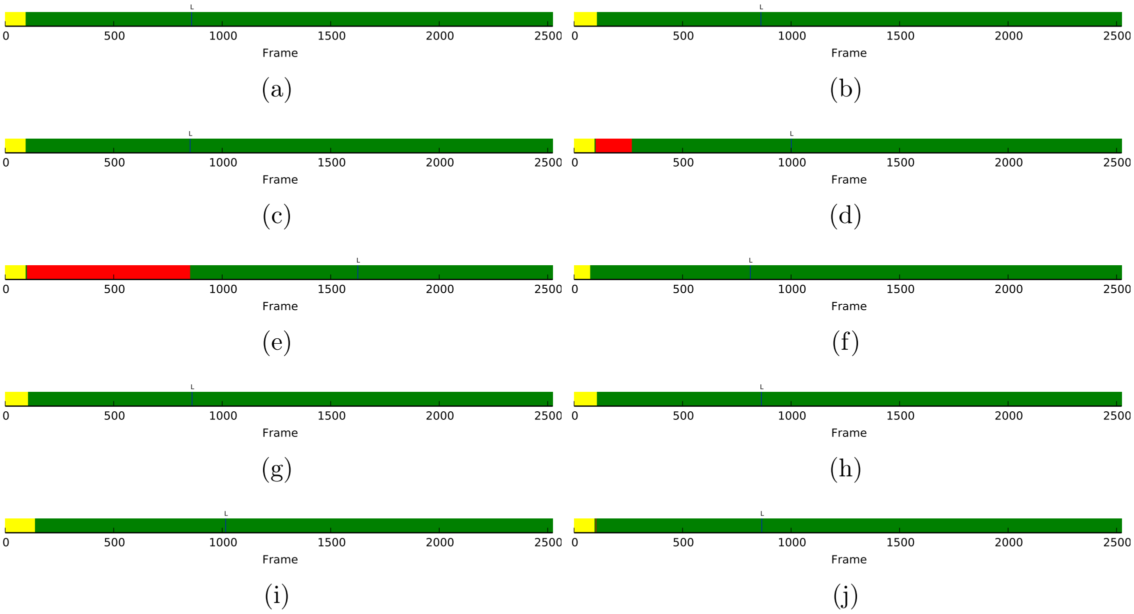
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 7: Dataset 7



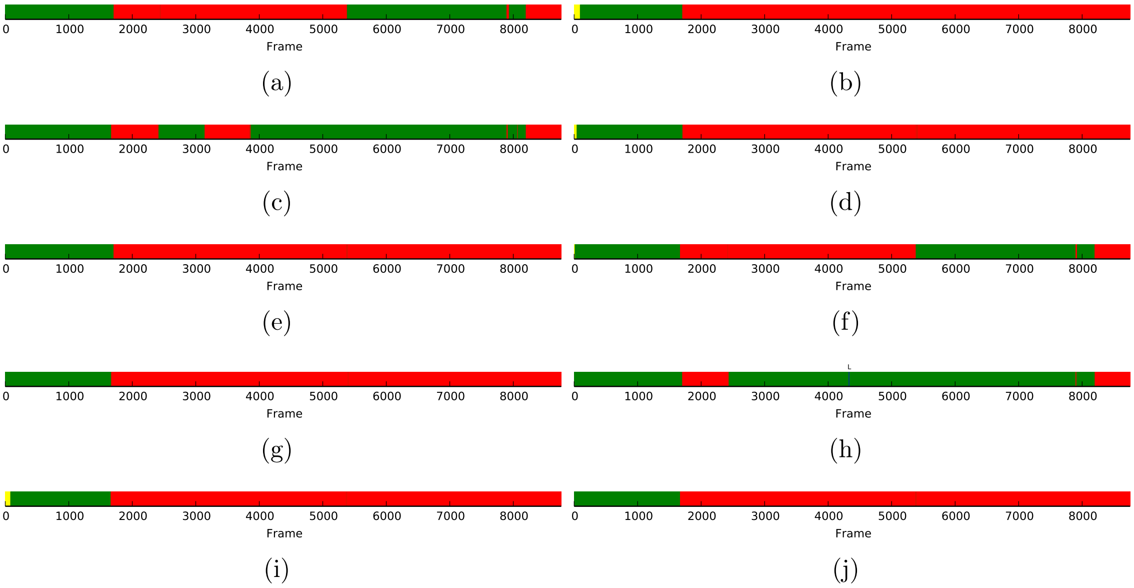
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 8: Dataset 8



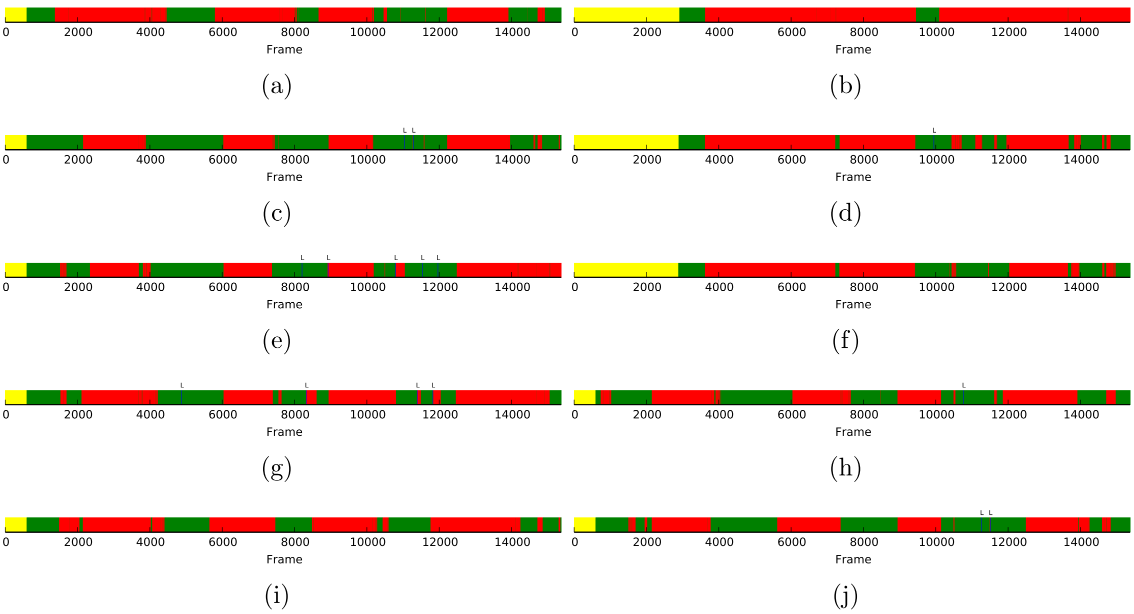
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 9: Dataset 9



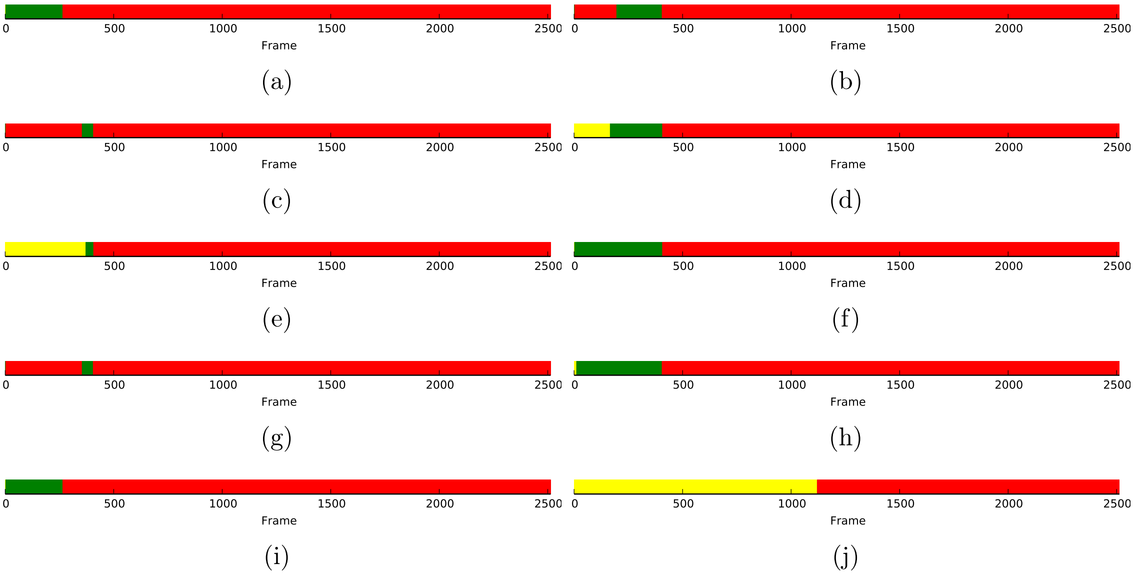
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 10: Dataset 10



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 11: Dataset 11



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 12: Dataset 12