Doctoral theses at NTNU, 2023:146

Marco Leonardi

Designing and Improving Techniques for Underwater Visual SLAM and Obstacle Avoidance

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Marco Leonardi

Designing and Improving Techniques for Underwater Visual SLAM and Obstacle Avoidance

Thesis for the Degree of Philosophiae Doctor

Trondheim, May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-5517-5 (printed ver.)
ISBN 978-82-326-6310-1 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2023:146

ITK No.: 2023-09-W

Printed by NTNU Grafisk senter

To everybody I had to say no because I had to work on this.

Abstract

This thesis focuses on designing and improving techniques for visual simultaneous localization and mapping (Visual SLAM or VSLAM) and obstacle avoidance, in the underwater environment.

From the perspective of underwater robotics, the underwater environment is rich in opportunities and dangers: examples of opportunities are climate change monitoring, flora and fauna mapping, biological studies and preservation, mining, and extraction of natural resources, and monitoring of underwater infrastructure, while threats are typically represented by obstacles, rough seas and currents, not-traversable fields, and marine fauna.

While sonars can provide information about physical elements present around the robot, they cannot capture color and semantic information, making it hard or impossible to achieve a certain level of autonomy. It becomes clear that the robot needs to perceive the world also through a camera.

Visual SLAM is the process of utilizing a camera to map the environment and at the same time localize itself in it, all of this in *soft real-time*. This is crucial as such information can be exploited to perform automatic re-routing and obstacle avoidance.

This thesis presents a stereo-camera-based obstacle avoidance field trial that demonstrates how a stereo camera, with proper active illumination, can be utilized instead of an acoustic-based sensor to perform active obstacle avoidance and then presents a series of methods to improve the robustness and performance of underwater VSLAM.

In particular, an emphasis is placed on monocular VSLAM, as monocular VSLAM is highly interesting in terms of robustness: a monocular VSLAM system can substitute a stereo VSLAM system in case of a single camera malfunction, it is superior in terms of compactness, as a single camera is easier and cheaper to place then two, and sometimes a single camera is the only available option for small robots.

This thesis presents a single image-only, standalone, and global method for robust loop closure detection, based on the encoded representation produced by a convolutional autoencoder. It also presents a method for keypoint rejection for feature-based VSLAM methods which avoid features to be detected on *unsuitable* surfaces, such as loose seaweed, marine fauna, and caustics. Finally, it presents a series of modifications to ORB-SLAM 2, one of the most successful feature-based monocular SLAM, in order to improve it for use in the underwater environment.

These modifications include a slight change to the initialization procedure, a

way to perform feature matching which yields a higher amount of valid matches, a partial synchronization between the front-end and the back-end, a procedure to detect station keeping without the need to know the scale of the movements and a pruning procedure which enables lifelong operations.

This thesis is edited as a collection of papers.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (Ph.D.) at the Norwegian University of Science and Technology (NTNU). The work was supported by the Research Council of Norway (RCN) founded Norwegian Centre of Excellence (SFF) NTNU AMOS - Centre for Autonomous Marine Operations and Systems (grant no. 223254). I have done this work at the Department of Engineering Cybernetics. My main supervisor has been Assoc. Prof. Annette Stahl, with co-supervisors Assoc. Prof. Edmund Brekke and Prof. Martin Ludvigsen.

List of Acronyms

ADCP Acoustic Doppler Current Profiler.
AKAZE Accelerated KAZE.
AMOS Center for Autonomous Marine Operations and Systems.
ASV Autonomous Surface Vehicle.
AUV Autonomous Underwater Vehicle.

BA Bundle Adjustment.
BFGS Broyden–Fletcher–Goldfarb–Shanno.
BoW Bag of Words.
BRIEF Binary Robust Independent Elementary Features.

CLAHE Contrast Limited Adaptive Histogram Equalization.
CNN Convolutional Neural Network.

DBoW Distributed Bag of Words.
DBSCAN Density-based spatial clustering of applications with noise.
DeMoN Depth and Motion Network for Learning Monocular Stereo.
DNN Deep Neural Network.
DSO Direct Sparse Odometry.
DVL Doppler Velocity Log.

ECEF Earth Centered Earth Fixed (frame).
ECI Earth Centered Internal (frame).
EKF Extended Kalman Filter.

GID Global Image Descriptor.
GNSS Global Navigation Satellite System.
GPS Global Positioning System.
GT Ground Truth.
GTSAM Georgia Tech Smoothing and Mapping.

HALOC Hash-based Loop Closure.

IMU Inertial Measurement System.
iSAM incremental Smoothing and Mapping.

KF Kalman Filter.

LBFGS Limited Broyden–Fletcher–Goldfarb–Shanno.
LDSO Direct Sparse Odometry with Loop Closure.
LIFT Learned Invariant Feature Transform.
LSTM Long Short-Term Memory.

MAP Maximum A Posteriori.

NN Neural Network.
NTNU Norwegian University of Science and Technology.

ORB Oriented FAST and Rotated BRIEF.

PTAM Parallel Tracking and Mapping.

RANSAC Random Sample Consensus.
ROV Remotely Operated Underwater Vehicle.

SAM Smoothing and Mapping.
SAS Synthetic Aperture Sonar.
SIFT Scale-Invariant Feature Transform.
SLAM Simultaneous Localization and Mapping.
SSE Streaming SIMD Extensions.
SURF Speeded Up Robust Features.
SVD Singular Value Decomposition.
SVO Fast Semi-Direct Monocular Visual Odometry.

TBS Trondheim Biologiske Stasjon, NTNU's center for marine biological research.
ToF Time of Flight (camera).

VO Visual Odometry.
VSLAM Visual Simultaneous Localization and Mapping.

List of Figures

Contents

Chapter 1

Introduction

Visual simultaneous localization and mapping (Visual SLAM or VSLAM) is the process of creating a representation of the environment using a camera (or a series of cameras), while, at the same time, determining the position of the camera, relative to the environment. VSLAM systems are in general composed of several algorithms and systems, often running concurrently, with the goal of *soft real-time* performance.

VSLAM is a very powerful tool when it comes to navigation and mapping: camera sensors are ubiquitous today, occupy a very small footprint, and use very little power. Such a sensor can produce a high-quality, dense, textured, and semantic 3D representation of the environment, no other sensors come close to such performance.

However, producing such a 3D representation of the environment requires light to be present in the scene or to be actively generated, images have to be taken often following a series of principles, like relative angles and scene overlap. In the underwater scenario, artificial illumination is often required, and dynamic elements are usually present in front of the camera in a variable quantity and in unpredictable directions and speeds, such as algae and marine/river fauna. Colors become also a function of the distance between the camera and the observed elements and additional image distortion is present (underwater visual perception challenges are further discussed in Section 2.4).

In this thesis, we focus firstly on reviewing a field trial where a stereo camera system is used to perform underwater obstacle avoidance, and then we proceed in exploring various aspects of monocular VSLAM, specifically focusing on loop closure, initialization, feature matching, synchronization of front-end and back-end, station keeping detection, and a pruning procedure, which enables lifelong operations.

It has to be noted that generating a VSLAM estimate often requires all the computational resources of the most powerful embedded platforms available today, in case *soft real-time* performance is required.

The purpose of this introductory chapter is to briefly outline the contributions of this thesis as well as to explain the background and motivation for this research.

1.1 Motivation and Problem Description

The underwater world still contains many secrets, and the exploration of this world is quite difficult and expensive. The vastness of the oceans, lakes, and rivers, the eventual presence of salt water, and the immense pressures to which underwater vehicles have to be subject are just a few examples of the challenges to overcome.

The prize for those who overcome such challenges is in part unknown, however, such a prize is likely to contain helpful information for monitoring and preserving the marine environment. There are indeed many threats that the marine environment is currently facing, such as oil spills, untreated sewage water, and endangered coral reefs.

Robotics will be absolutely crucial in this task: crewed missions bring with them high risks, ethical issues, and huge costs, as they involve situations that could be life-threatening. Robotics also help bring down the cost of the equipment, underwater robots don't have to carry all the life support systems that can be found in submarines. We have many underwater robots today which are able to perform a great number of tasks, but they require to be operated manually by an operator, and most of the time the operator has to be connected physically to the robot (called a *remotely operated vehicle*, ROV), due to salt water being an extremely efficient attenuator of electromagnetic waves.

The real turning point for ocean robotics is energy storage, harvesting, and/or generation development and autonomy. Due to the size of the oceans, it is realistic that underwater robot tasks could require from a couple of hours to years to be performed. Assuming the energy problem is solved, autonomy remains an issue. A robot should first deeply understand its task, its priorities, and capabilities, it should understand where it is located and how the environment looks around it, especially in the case of an unknown, unstructured environment, and it has to do it fast.

Cameras are a great help for autonomy. Contrary to sonar, they do provide color and semantic information and do provide more dense and continuous information about the environment, to enable true situational awareness for the robot. A camera on board an underwater robot can provide much more mission-sensitive information, especially when it comes to biological-oriented missions, for example studying algae and identifying fish species.

For a robot to navigate an environment that it has never seen before, it has at the same to map the environment and localize itself into it. As mentioned before, such a process is called simultaneous localization and mapping (SLAM).

The main work of this thesis focuses on understanding how successful camera-based SLAM, also called VSLAM, could be improved to better perform in *unstructured* (not previously known to the robot) underwater environments.

While SLAM underwater is mostly performed with acoustic sensors (due to the ability of sound to travel in the water much easier than light), the ability to perform SLAM underwater with cameras allows one to explore and navigate environments that are sensitive to acoustic pollution, like some natural environments. Acoustic-less navigation is also required for military operations, as any form of acoustic emission could help a hostile entity to identify the position of the emitter, to the point that submarine design has to take into account the submarine *acoustic*

signature [58] [26]. Acoustic sensors could also be too expensive for certain applications or impossible to deploy in small robotics platforms, while cameras can be accessed in a very wide variety of price ranges and sizes.

The underwater environment is tough for cameras: there is a problem in calibrating it, as there is another medium (typically a flat glass) in front of a camera, water can be turbid, and foreign objects or animals can pass in front of the camera, light can be not optimal and true colors are a function of the distance between the observed object and the camera. Working in such an environment requires the development of a series of robust algorithms which are designed to tackle the unpredictiveness of the sea and loss of visibility.

In this thesis, many challenges are explored and addressed, the methodology includes testing and adapting some of the most successful approaches and techniques to better tackle the underwater environment.

The methods developed in this thesis are evaluated on publicly available databases relevant to the community of underwater vision as well as datasets obtained during field trips involving ROVs and autonomous underwater vehicles (AUVs).

1.2 Research Questions

This thesis work is focused on exploring and tackling the challenges of underwater visual perception for underwater visual SLAM, with a particular emphasis on monocular camera systems, even though the investigation started at first with a stereo-camera system. Several research questions have been posed and answered during the research period:

- Is it possible to perform obstacle avoidance underwater using a stereo camera system?
  - Empirical tests performed in Paper A demonstrate underwater obstacle avoidance using solely a stereo camera system. The test involved many parties and the Candidate's contribution consisted in developing the software for tridimensional stereo reconstruction and integration with the control system. Different stereo-reconstruction approaches have been tested, and a novel (to the best knowledge of the Candidate at the time) enhancement to DBSCAN in order to reject noise is also presented.
- How can we improve the robustness of loop closure for monocular underwater visual SLAM?
  - This research question is addressed in Paper B, where a highly parallelizable, standalone algorithm for globally detecting loop closures using monocular images. Images are reduced to vectors through the encoding side of a convolutional autoencoder and are compared with each others utilizing the cosine distance. Such an approach is benchmarked against a bag-of-word approach and shows much higher precision in an underwater scenario. Limitations of this novel approach due to *perceptual aliasing* are presented to the reader. The Candidate is the author of all the work and research contributions related to Paper B.

- How can we improve the robustness of keypoint matching for monocular underwater visual SLAM applications?
  - In order to address this research question, a deep-learning based keypoint rejection system has been developed and is presented in paper Paper C. Rejecting keypoints that belong to *unsuitable surfaces* (i.e. moving surfaces) increases robustness by avoiding potential wrong initialization and tracking, outperforming approaches dedicated to dynamic environments. The Candidate is the author of most of the work related to Paper C, and he is responsible for all the research contributions related to it.
- How can we improve the best overall performing keypoint-based visual SLAM system for the underwater environment?
  - A series of ORB-SLAM enhancements have led to a system called Underwater Visual SLAM (UVS), presented in paper Paper D. Such enhancements touch almost all aspects of the original system and include specific improvements for underwater applications. UVS outperforms ORB-SLAM both in terms of median RMSE and loop closure quantity. The Candidate is the author of all the work and research contributions related to Paper D.

Research methodology encompasses empirical tests and experiments, as well as a theoretical test that has been evaluated on publicly available datasets.

The reason why this research has focused more on monocular cameras is that monocular camera research represents the minimal and fundamental stepping stone for visual SLAM, also monocular cameras have a series of advantages compared to stereo cameras:

- A lower overall power consumption, assuming computationally comparable algorithms are run for both stereo and monocular camera systems
- A higher resilience to hardware faults: a system that relies structurally on multiple cameras has more exposure to tail risk
- In the case of small robots, this could be the only choice

1.3 List of Contributions and Publications

In this section, the contributions are briefly presented in relation to the publications relevant to this thesis. We refer the reader to Chapter 4 and attached articles in Chapter 6.

Paper A: Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles

Authors: Marco Leonardi, Annette Stahl, Michele Gazzea, Martin Ludvigsen, Ida Rist-Christensen, Stein M. Nornes

Conference, Venue, Year: IEEE OCEANS Aberdeen, 2017.

Contributions: In a field test it was demonstrated how stereo vision can be used to perform obstacle avoidance underwater; A modification to DBSCAN

[45] for point clouds generated from a stereo vision system was presented, which makes the ϵ a function of the depth of the triangulated 3D points.

Paper B: Convolutional Autoencoder aided loop closure detection for monocular SLAM

Authors: Marco Leonardi, Annette Stahl

Conference proceedings, Year: IFAC-PapersOnLine, p. 159-164, 2018.

Contributions: A robust, highly parallelizable, standalone method for detecting globally loop closures was presented, based on a deep convolutional autoencoder.

Paper C: Deep learning based keypoint rejection system for underwater visual ego-motion estimation

Authors: Marco Leonardi, Luca Fiori, Annette Stahl

Conference proceedings, Year: IFAC-PapersOnLine 53.2 (2020): 9471-9477, 2020.

Contributions: A fast supervised way to generate a dataset for keypoint classification; A CNN-based, plug-and-play keypoint rejection system that rejects keypoints *unsuitable* for visual-ego motion estimation, in order to obtain more reliable estimates.

Paper D: "UVS - Underwater Visual SLAM: Robust Monocular Visual SLAM for Lifelong Underwater Operations"

Authors: Marco Leonardi, Annette Stahl, Edmund Brekke, Martin Ludvigsen

Journal, Year: Autonomous Robots, Springer, submitted 10/2020, accepted with minor revisions 11/01/2023

Contributions: A system including a series of modifications to ORB-SLAM 2 for underwater VSLAM was developed, called Underwater Visual SLAM (UVS): a three-view initialization procedure, which does not utilize a model selection procedure; a fast, exact descriptor-matching solution applied to the nearest neighbor search for triangulation, increasing tracking robustness; a model that predicts the camera pose in absence of visual information and map consistency; Partial synchronization between front-end and backend; a procedure to detect station-keeping operations while using the available computing power for performing global bundle adjustment; the same descriptor-matching solution applied for triangulating new points is applied for loop closures detection; a SLAM-friendly map and keyframes pruning procedure that enables lifelong operations

Other contributions in the overall context: Preliminary results which show how could be possible to estimate water salinity utilizing only an image of an object of known shape and size obtained underwater; a series of tests to determine the achievable performance running UVS on the embedded platform NVIDIA Jetson Nano platform leveraging the GPU; a series of tests

to determine the potential performance of UVS on sequences captured from Eelume's [39] underwater snake robot.

The research performed has contributed to the field of underwater vision by providing empirical experiments in underwater obstacle avoidance, and has contributed to the field of underwater visual SLAM by providing a series of standalone tools capable of improving robustness in underwater applications, as well as a new visual SLAM system based on ORB-SLAM that effectively overcomes many of the challenges of operating underwater.

The Candidate, Marco Leonardi stands as the first author of every publication, having personally and substantially originated the main contributions.

The research period lasted for four years. All publications were produced in that period. Papers A, B, and C were published during these years and Paper D was submitted at the end of that period. The journal manuscript was under review for more than two years and is currently in the minor revision stage ("conditionally accepted for publication").

1.4 Thesis Outline

The thesis is divided into seven chapters:

- In the first chapter motivation, background, methodology, challenges in underwater vision, and scientific contributions are introduced
- In the second chapter a review of sensors and platforms for underwater navigation is presented
- In the third chapter a review of VSLAM systems and underwater SLAM methodologies are presented
- In the fourth chapter the research performed by the Candidate is summarized
- In the fifth chapter the conclusions and recommendations for further work are discussed
- The sixth chapter contains all the original publications
- The seventh chapter contains additional documents which are related to the last original publication: Paper D: Underwater Visual SLAM system (UVS)

Chapter 2

Sensors and Perception for Underwater Navigation

In this chapter, we will take a look at underwater navigation and mapping techniques. We will conclude with an overview of how visual sensors fit in this scenario, and how established techniques can benefit from the addition of visual sensors.

2.1 Challenges of Underwater Sensing

The vast majority of the underwater navigation and mapping techniques today utilize acoustic sensors, the reason lies in the fact that acoustic waves propagate very well underwater, while electromagnetic waves don't, especially in saltwater (except electromagnetic waves of extremely low frequencies).

There are however a series of scenarios where performing navigation and mapping using a visual sensor is preferable, for example when utilizing small robots, where it is likely that fitting acoustic-based navigation and mapping sensors are prohibitive due to space constraints.

Cost is also a factor: cameras are now ubiquitous and high-quality sensors and optics can be found on the market for prices that are orders of magnitudes lower than acoustic systems.

Many acoustic sensors have a minimum clearance from the seabed, it could be those visual sensors are the only suitable ones for particular applications in shallow water. Acoustic sensors could impact natural life environments [162]. In applications where an underwater vehicle has to explore sensitive natural environments, it can be sensible to use visual sensors instead.

Finally, as acoustic information does not carry color information, cameras represent the only choice for applications that require 1:1 matching between 3D mapping and images. In such cases, visual navigation and mapping become a natural choice. An example of such an application is the studying and mapping of corals, where 3D information is insufficient in estimating the coral's health status [123].

While navigating underwater is not easy to benefit from global navigation satellite system (GNSS) navigation data, given the strong attenuation of electromagnetic waves underwater, a detachable specialized element of an underwater vehicle

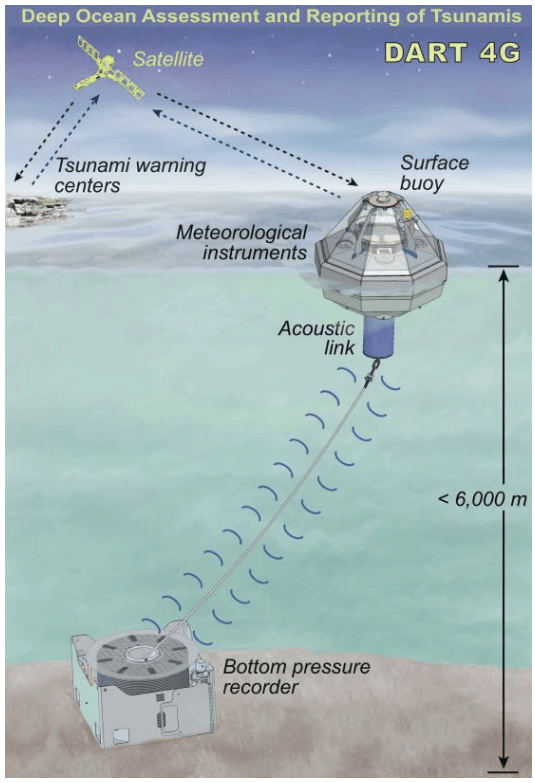

Figure 2.1: This figure shows the NOAA DART 4G system. This system is intended to detect tsunamis, and it uses satellite communication with a buoy. The communication of variations of pressures from the sensor stationary up to 6000 meters in the sea bottom is obtained through the use of an acoustic link. *Image courtesy of NOAA - Pacific Marine Environmental Laboratory in Seattle.*

could still be used for GNSS positioning, as this detached element could rise to the surface, and access the GNSS data, and could communicate with the underwater vehicle through a cable or acoustic communication. An example can be seen in [6], and in the National Oceanic and Atmospheric Administration (NOAA) Deep-ocean Assessment and Reporting of Tsunamis (DART), see Fig. 2.1.

2.2 Underwater environment in the light of hardware and sensors

The ocean represents the most unexplored part of our planet, less than 5% is explored according to the University of Hawaii at Manoa [121]. Exploring the oceans requires operating in a challenging environment, which is very hostile for human-made vehicles and robots. Surfaces exposed to water go through a process called *marine fouling*, which is the accumulation of various biological materials on these

surfaces, this, in turn, can damage the surfaces, increase drag, and obstruct sensors and movements of mechanical parts.

While pure water does not conduct electricity very well, mild contamination does dramatically increase conductivity, and so in any natural environment, water represents an immediate danger to any circuit, waterproofing is a critical component to deploy electronics onboard vehicles that will be in contact with water.

Operating underwater does bring the complications that the vehicle will be subjected to external pressure which does increase with its distance from the surface.

Salt water also accelerates oxidation processes (commonly called *rust*), increasing maintenance costs.

Underwater the electromagnetic waves-based human-made navigation and communication infrastructure of satellites and repeaters vanishes very fast, with the exception of Extremely Low Frequency (ELF) signals, which can carry a very small amount of information and require up to hundreds of kilometers long antennas [16]. Navigation and communication have then to rely mostly on acoustic devices, without a global reference system in unstructured environments.

2.3 Inertial and Magnetic Perception Sensors

Internal navigation relies on body-specific forces and angular rates. Compared to GNSS, inertial measurements do not provide a global position, but they are always available, as they do not rely on an external signal, which could also be scrambled. While even consumer-grade electronic GNSS systems can provide quite a precise position estimate, the quality of the measurements coming from an Inertial Measurement Unit (IMU) is highly varying between different sensors and is determined by the technological choices involved in their design and manufacturing, resulting in sensors available in very wide performance and price range. One of the principal characteristics of an IMU is the axis that it does provide:

- Three axes: in general it refers to the ability of the device to measure spacial accelerations
- Six axes: in general it refers to the ability of the device to measure spacial accelerations and angular velocities (also-called *gyroscope*, or more simply *gyro*)

Bias stability and scaling stability represent one of the key performance elements of accelerometers and gyroscopes [125], because biases cannot be observed without additional sensor readings.

In order to recover the relative position and orientation (attitude) that inertial measurement devices do provide, one could use the so-called *strapdown* navigation equations. These equations are called *strapdown*, as it refers to the way the IMU and the robot/vehicle are coupled together: the IMU has to be solidly attached to the robot/vehicle and it can be considered as a single rigid body.

Integrating the *strapdown* equations will provide position, velocity, and attitude, but errors contained in the measurements will accumulate over time, leading to the so-called *drift*.

The IMU which can be found in the market is advertised to have more than 6 axes, most commonly 9 and 10 axis. These devices are not properly only IMUs,

they contain a six-axis IMU, plus a 3-axis magnetometer (for the 9-axis), and a magnetometer plus a pressure sensor for the 10-axis.

A magnetometer measures the intensity of the magnetic field. As such, it can be used to locate the north pole (as a compass) in absence of significative magnetic disturbances. Magnetometer readings are also affected by time-varying biases and several potential production inaccuracies, like the sensor triad not being perfectly orthogonal or a different sensibility between the elements of the sensor triad.

2.4 Underwater Visual Perception and Camera Calibration

There are a series of additional challenges to be addressed when performing computer vision/visual perception underwater.

Computer vision is the science of extracting high-level concepts and information from images or videos, through the means of algorithms. Computer vision is a highly interdisciplinary science, which typically makes use of statistics, biology, geometry, and physics, as well as learning algorithms.

The first of the additional challenges for applying computer vision algorithms to underwater environment scenes is the non-linear absorption of electromagnetic waves from water [129][142], see Fig. 2.2.

As colors underwater are also a function of the distance of the observation, tasks that rely on colors, such as object recognition, pixel-to-pixel tracking, and keypoints matching are made more difficult. Direct visual odometry methods are at a particular disadvantage given the appearance-based approach (as will be experimentally demonstrated in Paper D: Underwater Visual SLAM system (UVS)), also they often require photometric calibration.

Light underwater is not only subject to absorption but also subject to reflection and refraction. As light is absorbed by water, any operation which is not in shallow water during a sunny day requires artificial illumination. Such illumination is likely to come from a location near the camera, and this could create reflections. Artificial illumination is also likely to illuminate particles suspended in the water, especially in natural environments, which do represent extra noise for a visual-based motion tracking system.

Water is not always clear, in both marine and freshwater environments there could be turbidity. Water turbidity does generally degrade the quality of the image. Turbidity could be generated by currents raising from the sea bottom various material or generated by the robot propelling system itself, as the current generated by the propellers could move the sea bottom. Specific studies have been performed trying to address keypoint matching in turbid water [57] [165].

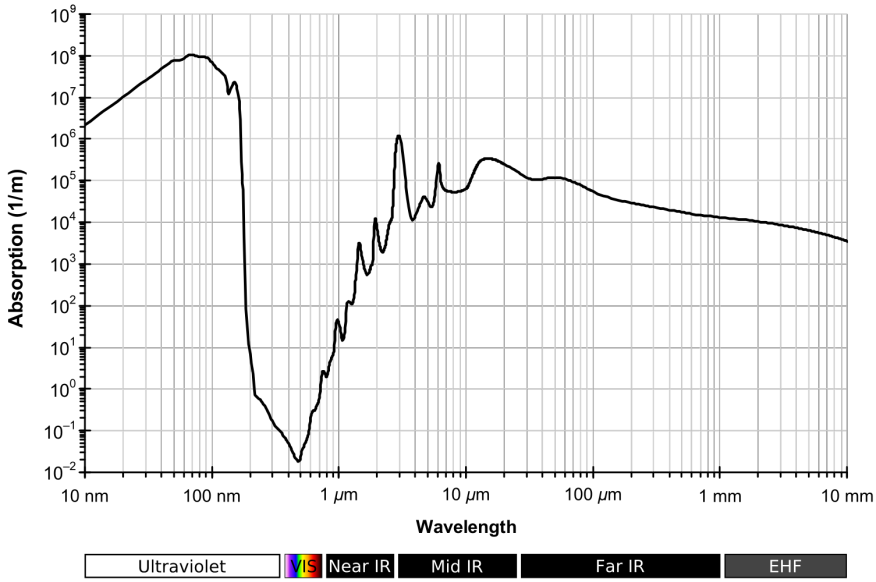

Figure 2.2: This figure illustrates how water absorbs electromagnetic radiation. Water absorbs warm colors like reds and oranges (known as long-wavelength light) and scatters the cooler colors (known as short-wavelength light). *Image courtesy of Wikipedia commons* [80].

Camera calibration does represent an extra challenge. Compared to in-air applications, the camera has to be kept away from the water, this implies that a protected medium is present between the camera and the camera's optic, called in the literature *port*, see Fig. 2.3.

Using a *dome port* does provide optical advantages, but the pinhole camera has to be positioned exactly in the focus center of the dome, which is very difficult. Some have investigated ways to correctly position a pinhole camera [143]. The more common instance of ports is represented by a *flat port*, which is represented by a flat transparent medium present in front of the camera, this kind of setup does not require the camera and the port to be specifically engineered for each other. The drawbacks of flat ports are added complexity in calibration, given the water-glass/acrylic-air refraction chain, which is not modeled in classic calibration procedures, also, while the refraction in the index of the flat port medium can be pre-calculated (or just looked up in refraction index tables), the one of the water cannot be easily known *apriori*, as it a function of salinity.

Performing the pinhole camera calibration with the *gold-standard method* [169] does work well only if both the calibration and the observations are done around the same distance from the camera. Research shows that does not need to be the case if a proper model is chosen [99].

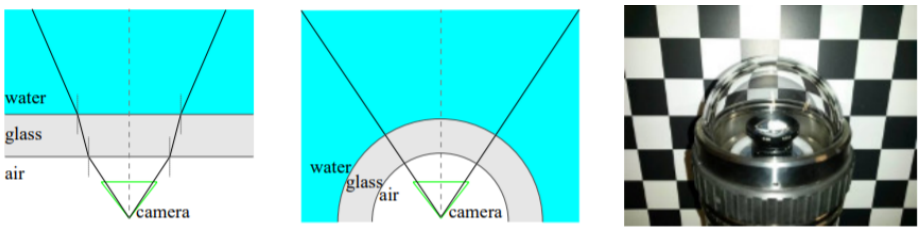

Figure 2.3: On the left, is a flat port, in the center is a dome port, and on the right is a picture of a dome port. In the dome port, in case the camera is correctly positioned, no light refraction has to be taken into account. *Image from article [143].*

Most visual odometry/SLAM systems assume that the environment is static, or at least primarily static so ego-motion can be estimated against the environment. In many cases, this assumption is violated in natural underwater environments.

Foreign objects could be represented by marine life passing in front of the camera, many underwater environments can be densely populated with marine life. One of the publications part of this thesis (Paper C, [90]) is dedicated to identifying dynamic objects not suitable for keypoint detection and matching for visual motion estimation.

Marine surfaces could be completely covered with vegetation that moves with water currents, violating the static environment hypothesis.

One of the few advantages that can be identified in operating cameras underwater is that given the relatively high density of water compared to air, we can assume that robots' accelerations will be smoother and less pronounced, and, as visual motion estimation requires overlapping frames, a lower framerate will be needed to satisfy the overlapping frames requirement, compared to, for example, drone-related applications.

Considering the tremendous challenges involved in exploring the oceans, robots are often placed first in line: the ability of robots to work almost continuously, and the ability to be resilient to extreme environmental factors, such as extreme pressure, temperature, and current, makes robots the perfect candidates for ocean exploration tasks. Historically, robotic platforms, like Remote Operated Vehicles, were chosen for underwater explorations. ROVs are connected through cables to an operative room on a boat or onshore and controlled by a human operator. The operator has access to a variety of measurement systems, such as cameras, sonars, and other sensors which can be present on the ROV (an example of ROV can be observed in Fig. 2.4). The operator can guide the ROV and manipulate the environment around it with robotic arms or dedicated tools fit for the specific task.

Unmanned vehicles which can be used to explore the oceans are divided into Autonomous Surface Vehicles (ASVs), gliders, and autonomous underwater vehicles (AUVs). Floating on the top of oceans there are ASVs, these vehicles are the ones that can provide month-long missions, they do have consistent access to GNSS and light, which can be used to recharge batteries in case the vehicle is fitted with solar panels. Being on top of the oceans ASVs are susceptibilities to oceanic waves and bad weather. Ocean gliders are hybrids between ASVs and AUVs, they operate

Figure 2.4: NTNU's ROV Minerva, belonging to NTNU's Applied Underwater Robotics Laboratory (AURlab) at Trondheim Biological Station (TBS). Author: Marco Leonardi.

most of the time on the surface, while they can dive and operate underwater when the mission requires.

In this variety of robotic platforms, several tasks can be identified which can be solved with the help of visual information.

While navigating underwater can be required to perform obstacle avoidance, in such cases, visual information can help identify the obstacle and determine alternative navigation paths [89] [37] [73].

In all the applications where a robot has to perform object manipulation underwater, having visual feedback to control the arm or actuators could be crucial, utilizing visual feedback to control the motion of the robot is called *visual servoing* [86] [55] [7].

A typical task for underwater vehicles consists of mapping the seabed [46] [63], this can be obtained through a technique called visual mosaicking [48] [120] [134]. Seabed mosaicking can also be performed with sonars [9] [108] [124], but sonars are unable to provide appearance information and fine-grained details which are accessible only through visual information.

Robots that are placed into an unstructured underwater environment and are

requested to navigate it, are usually required to perform SLAM. Underwater SLAM can be performed with both acoustic [133] [145] [168] and visual inputs [140] [81] [127], with the main difference that, exactly like for seabed mosaicking, acoustic information does not carry appearance information. Acoustic and visual information can be combined to achieve higher quality SLAM estimates [131].

Chapter 3

A Review of Simultaneous Localization and Mapping Systems

In this chapter a review of Visual SLAM and the relevant approaches is presented, together with a review of Underwater-specific SLAM methodologies, while examining the available sensors specific to the underwater domain.

3.1 Visual SLAM

Visual SLAM is a form of SLAM where one or multiple cameras are used as sensors. In the literature, VSLAM is a term that is used also for non-conventional cameras, like time-of-flight (ToF), structured light, and other kinds of cameras.

3.1.1 Theory

Here are briefly introduced a series of mathematical tools which are important to facilitate the understanding of Visual SLAM methods.

Corner features

For *features extraction* is intended the computer vision process of identifying a series of salient elements in a digital image. The goal of such a process is, in general, the ability to re-identify such features in other images and so be able to establish a relationship between two or multiple images.

The most popular form of feature is the corner feature, which can be formally identified as a region where two edges intersect.

Corner detection is one of the oldest problems in computer vision and so the literature on this subject is really vast, here is a list of the historically most popular methods: SIFT [98], Harris corner detector [67], Shi-Tomasi [144], KAZE [5], FAST [153] and ORB [139].

Harris corner detector is calculated using the image derivatives, by considering the Taylor expansion of the *sum of squared differences* (SSD) of two image patches from a grayscale image, in order to build a structure tensor, the eigenvalues of the structure tensor are then used to calculate a response called the *Harris response*:

$$R_{Harris} = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \qquad 0.06 \geq k \geq 0.04 \tag{3.1}$$

Where R_{Harris} represents the Harris response, λ_1 and λ_2 are the two eigenvalues of the structure tensor, and k is an empirically determined constant. The higher the response, the higher the chance of a corner. The Harris response is perfectly rotation invariant, given that eigenvalues magnitude is rotation invariant.

Shi-Tomasi is an improvement over the Harris corner detector, the response function is slightly modified [76]

$$R_{Shi-Tomasi} = min(\lambda_1, \lambda_2) \tag{3.2}$$

Where $R_{Shi-Tomasi}$ is the Shi-Tomasi response and λ_1 and λ_2 are the two eigenvalues of the structure tensor. Shi-Tomasi provided empirically better results [76], the reason is that keypoints selected in this way are numerically well conditioned for estimating image displacement [172].

KAZE (Japanese word for *wind*) applies nonlinear diffusion filtering [159] [135], in order to offset multi-scale Gaussian-based blurring drawback: the reduction in localization accuracy [5], as Gaussian blur does smooth in the same manner the noise and the details. The computational performance of KAZE is pretty poor, as the authors point out is comparable to SIFT, while the repeatability performance is superior to both SIFT and SURF in all the tests, which include image blur, JPEG compression, rotation, zoom and viewpoint change. A less computationally demanding version of KAZE has been published, with the name of *accelerated KAZE* (AKAZE) [4]. AKAZE utilizes the so-called *Fast Explicit Diffusion* (FED) [160] to speedup feature detection in the non-linear scale space, plus they utilize a modified *Local Difference Binary* [166] that exploits the gradient information already present in the non-linear scale space.

Features from Accelerated and Segments Test or simply FAST [138], is a methodology for finding corner features in a very inexpensive way, compared to many other methods. FAST defines a so-called *segment test detector*: it considers the pixels belonging to a Bresenham's circle of radius 3, centered in a candidate corner, in order to establish if it's a corner or not. In case n contiguous pixel has a higher, or lower intensity (plus a threshold), than the candidate corner intensity, then the candidate might be a corner. A subsequent machine learning approach based on information content is used to create a decision tree that approximates the segment test detector.

Oriented FAST and Rotate BRIEF (ORB) [139], represents one of the most successful combinations of keypoint detector and descriptor, due to its ability to

perform similarly to SIFT and be two orders of magnitudes faster. ORB utilizes the FAST corner detector but adds crucial information: orientation. The corner orientation is calculated starting from Rosin's *intensity centroid* [137], however, they ignore if a corner is dark or bright. Rosin defines the moment m of a patch as follows:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{3.3}$$

Where p and q are the order of the moments, and $I(x, y)$ represents the intensity of the image at location (x, y). Considering $x, y \in [0, 1]$, the centroid is defined as:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \tag{3.4}$$

The orientation of the patch is then:

$$\theta = atan2(m_{01}, m_{10}). \tag{3.5}$$

Descriptors

In order to match features between each other a form of description is needed, such description is provided by so-called *descriptors*. Descriptors are in general algorithms which receive a corner as input and attempt to provide a feature vector that will uniquely identify the corner, in such a way that it is as invariant as possible to illumination, change of relative angle of observation, and changes in scale, typically keeping an eye on its computational complexity. There might be thousands of corner features in an image, and if each of them has to be calculated a descriptor, such an operation really benefits from maintaining low computational complexity.

One of the most popular early keypoint detectors and descriptors is called Scale-invariant feature transform (SIFT) [98]. Its popularity derives from its robustness, which derives from its extensive use of derivatives: intuitively, image derivatives are quite invariant to illumination and scale changes. The SIFT descriptor is built by taking a 16x16 window around the keypoint, and dividing it into 16 sub-blocks of 4x4, and for each of these blocks an orientation histogram is created, these orientation histograms represent the descriptor.

Noticing that SIFT keypoints have a location, a scale, and an orientation, in order to achieve rotation independence of the feature vector, the rotation of the keypoint is removed from the feature vector, and to achieve illumination independence, the feature vector is normalized. Scale invariance is obtained not through the descriptor, but in the way the keypoint is found: the keypoint is calculated in a

scale space where the keypoint would produce the highest response, so at changes in scale in the image space, the keypoint would be found in the same scale space and so produce the same descriptor.

Binary descriptors represent a different approach to descriptors. The ability to have a descriptor that is in a binary form unlocks the ability to calculate the distance between two descriptors using the Hamming distance, which is orders of magnitude faster than calculating the L2 distance, especially when it's possible to utilize CPU or GPU-specific low-level instructions. An example of a binary descriptor is Binary Robust Independent Elementary Features (BRIEF).

BRIEF defines the following test τ for an image patch p which is smoothed using a Gaussian kernel:

$$\tau(p; x, y) \coloneqq \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Where $p(x)$ represents the intensity at coordinate $x = (u, v)$. Let's assume there is a set n_d of (x, y) locations for each patch where this test is performed, the n_d BRIEF descriptor is defined as:

$$f_{n_d}(p) \coloneqq \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{3.7}$$

BRIEF fails to be useful in the case of even small image rotations (see Fig. 7 [139]), for such reason the authors of ORB utilize a so-called *rotated BRIEF* (rBRIEF). The solution consists in learning a *good* set of binary tests to perform, the ones which have high variance and are uncorrelated over a large dataset.

Kalman Filter

A Kalman Filter is a two-step algorithm whose goal is to produce an estimate for a set of variables (called *state*) and its covariance. The Kalman filter is an optimal filter, in the sense that it provides the minimum possible mean square error (MSE) error on the state estimate. The optimality of the Kalman filter holds only the uncertainty of the state, as well as process noise and measurement noise can be described by a Gaussian distribution, and the system dynamic has to be linear and time-invariant [79], in addition, the process noise and measurement noise have to be white noise. The first step is called *prediction*: in this step, the current state estimate, as well as the covariance are predicted according to known external influences and the system dynamics:

$$\begin{aligned} \hat{x}_k &= F_k \hat{x}_{k-1} + B_k u_k \\ P_k &= F_k P_{k-1} {F_k}^T + Q_k \end{aligned} \tag{3.8}$$

Where x represents the state, P the covariance matrix of the state, the notation $\hat{\cdot}$ indicates an estimate, u represents the known external influences, B the matrix which relates the known external influences to the current state, Q the matrix which represents additional uncertainty given the environment where the state is estimated in.

The second step is called *update*: in this step, the estimates from the prediction step are refined with the measurements coming from a series of sensors. The refinements key step consists in multiplying the Gaussian distribution obtained by estimating the sensor readings we are expecting to see with the actual sensor readings distribution:

$$\begin{aligned} \hat{x}_k' &= \hat{x}_k + K(z_k - H_k\hat{x}_k) \\ P_k{}' &= P_k - KH_kP_k \\ K &= P_kH_k{}^{\mathsf{T}}(H_kP_kH_k{}^{\mathsf{T}} + R_k)^{-1} \end{aligned} \tag{3.9}$$

Where H is the sensor matrix, R the covariance which represents the uncertainty of the sensor measurements, z represents the sensor readings and K is the so-called *Kalman gain*, the matrix form of a common term present in both the mean and the variance equations resulting from multiplying two Gaussian distributions.

The linearity of the system dynamics excludes the possibility to apply the Kalman filter to many real-world problems. In order to address this issue and apply the Kalman filter to a much larger class of problems, the so-called EKF (*extended Kalman filter*) has been developed. The difference between the KF and the EKF is that in the EKF the system can be non-linear, but must be differentiable, as the whole idea of the EKF can be summarized as a linearization around the current estimate, using Taylor expansion. EKF loses all the optimality properties of the KF, there is no more guarantee of convergence, and it might be very sensitive to initialization, depending on the nature of the nonlinearities.

A typical use of an Extended Kalman Filter that has been popular in the past in Visual SLAM has been to estimate the state of the camera (position and attitude) together with all the positions of all the features, an example is MonoSLAM [33].

Kalman Filters have been thoroughly studied and adapted to specific problems, two examples of specialized Kalman Filters are the *Unscented Kalman Filter*, which provides better estimates in case F and H are highly non-linear, and the *Ensemble Kalman Filter*, suitable for high-dimensional problems.

Maximum a posteriori estimation

The so-called MAP (maximum a posteriori) estimation is the estimation of the mode of a posterior distribution. MAP is the generalization of the popular *maximum likelihood* estimation, where there is no assumption about the prior distribution of the parameters object of the estimation:

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}}\, f(x|\theta)g(\theta) \tag{3.10}$$

Where the notation $\hat{\cdot}$ represents an estimate, $\operatorname{argmax}_\theta$ the notation for the operation that will find the maximum value of θ, θ represent the vector of the parameters, $f(x|\theta)$ the posterior distribution and $g(\theta)$ the prior of the parameters.

MAP estimates can be obtained analytically as long as the mode(s) of the posterior distribution are given in a closed form, this is unlikely to be the case in computer vision-related estimation problems.

A more typical way to obtain MAP estimates is using numerical optimization methods such as gradient-based methods, or alternatively, an iterative method such as expected-maximization [91] or Monte Carlo methods and genetic algorithms.

Fundamental, Essential, Homography Matrix Estimation

In the field of computer vision, more precisely in *epipolar geometry* (the geometry of stereo vision), a *Fundamental matrix* or *bifocal tensor* is a matrix that relates two corresponding homogeneous image coordinates. Defining x and x' two corresponding homogeneous image coordinates, the Fundamental matrix F expresses the following relationship:

$$x'^T F x = 0 \qquad F_{3\times 3}\ ,\ rank(F) = 2 \tag{3.11}$$

The estimation of the Fundamental matrix is a well-studied problem and many algorithms have been published in order to estimate it. In general, such an operation requires a set amount of image correspondences to be found in order to proceed to the estimation, however, as for the vast majority of all computer vision tasks, deep learning methods have been developed [132], which requires no correspondences.

Traditional methods are the so-called *7-point algorithm* and the *8-point algorithm* [70].

Each correspondence provides a linear equation, which it's possible to write in the form:

$$Af = 0. \tag{3.12}$$

It turns out that A has only 7 degrees of freedom given the fact that $det(A) = 0$ and is determined up to scale [69] [31]. The problem is that with only 7 points there are three possible solutions.

Using instead 8 points, a single solution can be found, as we only have linear equations, in the same number as the unknowns.

When it comes to practical estimations, given possible wrong and imprecise image correspondences, more than 8 correspondences are provided to a Random

sample consensus (RANSAC) framework [68], which internally utilizes 7 points (as with 7 points there are fewer chances to incorporate noise in the chosen solution).

To the best knowledge of the author of this thesis, the minimal solver for the Fundamental matrix today is a two-point solver [15], which provides a solution as accurate as the 8-point algorithm.

Closely related to the Fundamental matrix, there is the *Essential* matrix E, first introduced by Longuet-Higgins [95]. The Essential matrix can be obtained from the Fundamental matrix, by the mean of the *Intrinsic* matrix K:

$$E = K'FK^T \tag{3.13}$$

It is to be noted that in case the two views are related by F originated from the same camera, then $K' = K$. The Intrinsic matrix K is a 3×3 matrix that contains the focal length and principal point of a camera modeled accordingly to a pin-hole camera model.

Exactly like the Fundamental matrix, the Essential matrix relates two corresponding homogeneous image coordinates:

$$x'^T Ex = 0 \qquad E_{3\times 3}\ ,\ rank(E) = 2 \tag{3.14}$$

The Essential matrix can be decomposed in relative camera rotation and translation, up to scale, so this implies that the Essential matrix contains only five unknowns. This property of the Essential matrix has been exploited by Nistér [119] in combination with RANSAC in order to obtain a minimal solver for the Essential matrix for practical uses. First has to be considered **Theorem 1** of [119]: *A real non-zero matrix* 3×3 E *is an Essential matrix if and only if*:

$$EE^T E - \frac{1}{2} trace(EE^T)E = 0. \tag{3.15}$$

This equation results in a cubic constraint later used to recover the Essential Matrix. Each of the five correspondences gives a constraint in the form:

$$\tilde{q}^T \tilde{E} = 0, \tag{3.16}$$

where

$$\begin{aligned} \tilde{q} \equiv & [x_1x'_1\ x_2x'_1\ x_3x'_1\ \ x_1x'_2\ x_2x'_2\ x_3x'_2\ \ x_1x'_3\ x_2x'_3\ x_3x'_3]^T \\ \tilde{E} \equiv & [E_{11}E_{12}E_{13}E_{21}E_{22}E_{23}E_{31}E_{32}E_{33}]^T. \end{aligned} \tag{3.17}$$

By stacking five vectors $\tilde{q}^T$, one for each correspondence, a matrix of dimension 5×9 is obtained. The goal is then to compute the four vectors $\tilde{B}, \tilde{Y}, \tilde{Z}, \tilde{W}$ that span the right nullspace of this matrix, and methods such as singular value decomposition (SVD) or QR factorization can be used. The four vectors $\tilde{B}, \tilde{Y}, \tilde{Z}, \tilde{W}$ correspond to four 3×3 matrices B, Y, Z, W, and so the essential matrix can be written in relation to these matrices as

$$E = bB + yY + zZ + wW. \tag{3.18}$$

With w set conveniently equal to 1, the four scalars are defined up to scale. Substituting 3.18 into 3.15 and then performing Gauss-Jordan with partial pivoting and defining a set of four additional equations, a set of five equations is obtained, in the attempt to find the null space of such system, a tenth-degree polynomial is obtained, obtaining the root of such polynomial yields the Essential matrix. The roots can be found using Sturm-sequences to bracket the roots, followed by a root-polishing scheme [119].

Unfortunately, finding the solution to the ten-degree polynomial could potentially be ill-conditioned [14], this leads to the development of a an iterative five-point algorithm [100].

More recently a non-minimal Essential matrix solver has proven to be able to converge to the globally optimal solution [170].

With the goal to estimate the relative camera rotation and translation between two images, there is an alternative to Fundamental/Essential matrices: the Homography matrix.

The Homography matrix is a 3×3 matrix that represents a bijective isomorphism and is related to the transformation between two planes. Exactly like an Essential matrix, a Homography matrix can analytically [102] recover rotation and transition (up to scale) between two images.

There are only four point correspondences needed to estimate a homography, but unfortunately, these four points have many requirements: at least three points don't have to be collinear, they all need to be co-planar and the sample should consist of points with a good spatial distribution over the image [78].

Classical error functions which are utilized to estimate a homography H are to use the so-called *symmetric transfer error*:

$$sm_{error} = d(x, H^{-1}x')^2 + d(x', Hx)^2 \tag{3.19}$$

Or the classical *reprojection error*, also called *geometric error*:

$$reProj_{error} = d(x, \hat{x})^2 + d(x', \hat{x}')^2 \tag{3.20}$$

Where $\hat{x}' = H\hat{x}$ represents the perfect correspondence. Another option is Sampson's error (the first-order approximation of the geometric error).

The problem of Homography estimation has also been approached using deep learning techniques demonstrating the ability to outperform traditional methods in particular scenarios as well as high flexibility, and robustness to light changes [36] [118].

Bundle adjustment

Bundle adjustment (BA) in computer vision is the process of simultaneously refining through optimization of tridimensional coordinates, the camera motion, and the parameters of the model utilized to represent the camera.

In the VSLAM literature, the definition is slightly different, in general, BA does not include refinement of the camera parameters until explicitly mentioned. Furthermore, the following terms can be often encountered:

- *motion-only* BA: Bundle Adjustment where only the camera poses are being optimized.
- *structure-only* BA: Bundle Adjustment where only the 3D points are being optimized.
- *local* BA: Bundle Adjustment that encompasses a part of all the available camera poses and 3D points.
- *full* BA: Bundle Adjustment that encompasses all the available camera poses and 3D points. It could also mean that both 3D points and camera motion are refined when used in opposition to structure-only and motion-only BA, but with a limited selection of poses and map points.

Formally, bundle adjustment can be defined as the following minimization problem [78]:

$$\min_{\hat{P}^i, \hat{X}_j} \sum_{ij} d(\hat{P}^i \hat{X}_j, x_j^i)^2 \tag{3.21}$$

Where X represents a set of 3D points, and P represents a set of camera matrices, where $P^i = K_i[R_i|t_i]$.

The most often encountered method is typically a sparse Levenberg-Marquardt [107], however, research has shown there are other options that might be more suited [97], like a sparse version of Powell's dog leg non-linear least squares [130]. Bundle adjustment can also be computed using multiple threads [163], making it possible to perform the desired optimization over a large amount of camera poses and 3D points. Deep networks can also be specialized to deal with the BA [151].

Pose graph optimization

Many problems in computer vision and robotics can be solved through least squares optimization, with a cost function in form of a graph [85]: the reason is that a

variable that we are looking to determine is dependent only on the value itself, as well as as the position of the sensor. An example of such a case that applies to computer vision is the determination of a tridimensional point world position, it depends only on its location and the location of the camera that is observing it. All the time that a problem can be formulated as a set of pairwise observations, it can be formulated as a graph, where the nodes are the variables to determine and the edges between two nodes represent pairwise observations. Superior efficiency can be obtained when compared to Gauss-Newton, Levenberg-Marquardt, and other similar methods, the reason is the following:

- Such graphs are naturally sparse: there is no need to infer relationships between variables, they are built together with the graph
- The formulation of the problem in a graph form allows the application of several different methods which can greatly increase computational efficiency

Formally, the generic problem that can be solved in the least-square sense of many robotics and computer vision problems using a graph formulation can be so written [85] as

$$\begin{aligned} x^* &= \underset{x}{\operatorname{argmin}} F(x) \\ F(x) &= \sum_{\langle i,j \rangle \in C} (\underbrace{e(x_i, x_j, z_{ij})^T \Omega_{ij} e(x_i, x_j, z_{ij})}_{F_{ij}}) \end{aligned} \tag{3.22}$$

Where x is a vector of parameters where each element represents a generic parameter block, x^* the estimated optimal solution, z_{ij} and Ω_{ij} represent respectively the mean and the information matrix of a constraint relating the parameters x_i and x_j, and $e(x_i, x_j, z_{ij})$ is a vector error function that measures how well x_i and x_j satisfy the constraint z_{ij}. The problem in the equation 3.22 can be represented as a directed graph: each parameter block x_i can be represented as a node and the ordered constraints can be represented as the edges between these nodes.

Classic methods for solving multivariate functions minimization solve the problem by performing a first-order linearization around a good initial estimate and calculating appropriate increments:

$$x^* = \breve{x} + \Delta x^* \tag{3.23}$$

Where $\breve{x}$ is the initial guess of a system in such form:

$$H \Delta x^* = -b \tag{3.24}$$

Such methods assume that the space of x is Euclidean and such an assumption cannot be made for many robotic and SLAM-related problems, as usually, these

problems require finding an attitude and rotations belonging to $SO(2)$ or $SO(3)$. A typical way to go around this limitation is to express the increments Δx_i in a different space from one of the parameters block x [85].

One of the most successful methods for performing pose graph optimization is g2o or g^2o [85]. The authors of g^2o define a new error function where the increments Δx_i are perturbations around the current variable $\breve{x}_i$. The increments Δx_i use a minimal representation for the rotations (like Euler angles), while x_i utilizes an over-parametrized one (for example rotation matrices or quaternion). Defining the non-linear operator $\boxplus : Dom(x_i) \times Dom(\Delta x_i) \rightarrow Dom(x_i)$, the optimized variables can be obtained with the following:

$$x_i^* = \breve{x} \boxplus \Delta x_i^* \tag{3.25}$$

In tridimensional SLAM the increments Δx_i can be represented with a translation vector and with the axis of a quaternion (normalized), also the operator $\boxplus$ can be substituted with the motion composition operator $\bigoplus$ [146]. The error function can be rewritten as follow:

$$\begin{aligned} e_{ij}(\Delta x_i, \Delta x_j) &\overset{\text{def.}}{=} e_{ij}(\breve{x}_i \boxplus \Delta x_i, \breve{x}_j \boxplus \Delta x_j) \\ &= e_{ij}(\breve{x} \boxplus \Delta x) \simeq e_{ij} + J_{ij} \Delta x \end{aligned} \tag{3.26}$$

Where $\breve{x}$ spans over the original over-parametrized space. The Jacobian J_{ij} is then:

$$J_{ij} = \left. \frac{\partial e_{ij}(\breve{x} \boxplus \Delta x)}{\partial \Delta x} \right|_{\Delta x = 0} \tag{3.27}$$

The increments $\Delta \hat{x}^*$ need to be remapped to the original over-parametrized space, as they are computed in the local Euclidean neighbor of $\breve{x}$ (the initial guess).

Certain problems, like bundle adjustment, result in an H matrix 3.24 that, with just a reordering of the variables in such a way that the camera poses are at the top, is possible to obtain further performance increase:

$$\begin{pmatrix} H_{pp} & H_{pl} \\ H_{pl} & H_{ll} \end{pmatrix} \begin{pmatrix} \Delta x_p^* \\ \Delta x_l^* \end{pmatrix} = \begin{pmatrix} -b_p \\ -b_l \end{pmatrix} \tag{3.28}$$

Where the subscript p stands for poses and l stands for landmarks. This form allows formulating an equivalent reduced system by taking the Shur complement

of H [52]. Solving this system allows recovering the increments Δx_p^*, which then can be used to find the increments for the landmarks Δx_l^*.

Georgia Tech Smoothing and Mapping (GTSAM) [34] is an approach to pose-graph optimization based on factor graphs [94]. Factor graphs are derived from a special case of *hidden Markov model* (HMM): considering the measurements known, the posterior probability of the states we would like to estimate, given the measurements is simply given by the product of *factors*, some of which derive from the Markov chain, some other which derive from likelihoods defined in the form $L(X_t; z)$, can be obtained through the multiplication of *factors*:

$$\begin{aligned} &P(X_0, X_1, ..., X_n \mid Z_0, Z_1, ..., Z_n) \propto \\ &P(X_0)P(X_1|X_0) \ ... \ P(X_n|P_{n-1})L(X_0; z_0) \ ... \ L(X_n; z_n) \end{aligned} \tag{3.29}$$

Similarly to g^2o, GTSAM utilized Levenberg-Marquardt and conjugate gradient optimizer, Gauss-Newton optimizer, Powell's dogleg, however it also utilizes *incremental Smoothing and Mapping*(iSAM)[77].

The approach called iSAM is derived from the previous work called *square root SAM* [35]. Following the probabilistic factor graph formulation of SLAM see in equation 3.29, the joint probability of all variables and measurements present in SLAM is:

$$P(X, L, Y, Z) \propto P(x_0) \prod_{i=1}^{M} P(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} P(z_k|x_{i_k}, l_{j_k}) \tag{3.30}$$

Where $X = x_i, i \in 0....M$ are the robots states, $P(x_0)$ is the prior of the initial state, $L = l_j, j \in 1...N$ the landmarks, the control inputs $U = u_i$ and the the landmark measurements $Z = z_k, k \in 1...K$, $P(x_i|x_{i-1}, u_i)$ the motion model and $P(z_k|x_{i_k}, l_{j_k})$ the landmark measurements model, with x_i and z_k affected by zero-mean Gaussian noise. Estimating the states and the landmarks can be done by converting the problem into a least-squares estimation problem, by minimizing the negative log of the joint probability:

$$\begin{aligned} X^*, L^* &= \underset{X,L}{\operatorname{argmax}} P(X, L, Y, Z) \\ &= \underset{X,L}{\operatorname{argmin}} -logP(X, L, Y, Z) \\ &= \underset{X,L}{\operatorname{argmin}} \left\{ \sum_{i=1}^{M} \|f_i(x_{i-1}, u_i) - x_i\|^2_{\Lambda_i} + \sum_{i=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Gamma_k} \right\} \end{aligned} \tag{3.31}$$

Where $f_i(x_{i-1}, u_i)$ represents the robot state estimated through the process model, $h_k(x_{i_k}, l_{j_k})$ represents the estimates from the measurements model, the

norm operator represents the Mahalanobis distance, Λ_i and Γ_k represent respectively the covariances of the process and measurement noise. Performing linearization of the measurement equations leads to a sparse Jacobian and by collecting all the variables a standard least square problem can be constructed.

The standard least square problem can be solved, for example with Cholesky decomposition or with QR factorization, iSAM utilizes the latter approach and the so-called *Given rotations* [156] approach. At the arrival of new measurements, instead of performing a new factorization, iSAM performs a *QR factorization update*, which results in much higher computational efficiency.

Loop closures unfortunately are prone to attack the sparsity of the R matrix, iSAM utilizes COLAMD [32], an efficient heuristic to re-order the matrix R, improving its sparsity.

In iSAM data association is solved by utilizing the *maximum likelihood* data association criteria [13] and the Jonker-Volgenant-Castanon (JVC) assignment algorithm [74].

Other than a great computational efficiency, iSAM provides Gaussian uncertainty estimates for the landmarks and the robot positions, which can be useful in other aspects of SLAM, such as refining sensor-specific measurements.

Utilized in production by Google since 2010, Ceres [2] is a non-linear least square solver, which does not rely on the sparsity of the underlying problem, does not require derivatives to be supplied as it performs numerical differentiation, the loss function can be regularized and for non-euclidean spaces, the user can specify the geometry of the local tangent space. Ceres provides two different main solvers for two different kinds of problems:

- Trust region solvers: these kinds of solvers are for problems that require high accuracy, as optimizers are available Levenberg-Marquardt, Powell's Dogleg and Subspace Dogleg methods [20] [19] and as for linear solvers dense QR factorization, dense Cholesky factorization and sparse Cholesky factorization custom Schur complement based dense, sparse, and iterative linear solvers.
- Line search solvers: for very large problems where computational complexity becomes a problem, line search based comes to help, Ceres offers several variants of non-linear conjugate gradients, Broyden–Fletcher–Goldfarb–Shanno (BFGS) and Limited-memory BFGS (LBFGS).

One of the most modern alternatives, SE-Sync [136] re-formulate pose-graph optimization as a *synchronization* problem over $SE(d)$, i.e.: estimate the values of a set of unknown group elements $x_1, ..., x_n \in SE(d)$ given noisy measurements of a sub-set of their pairwise relative transforms $x_i^{-1} x_j$. SE-Sync is able to recover a *certifiably* [12] globally optimal solution to the SE synchronization problem in a non-adversarial noise regime, and it is able to achieve this by utilizing a semidefinite relaxation of the MLE which will provide an exact MLE estimation, as long as the measurement noise is below a certain threshold, with the great property of being able to verify a posteriori that the previously mentioned threshold has not been achieved, and so certifying. Experimental evaluations have demonstrated that SE-Sync is able to recover certifiably correct globally optimal solutions where the solution is affected by noise an order of magnitude higher than the one typically encountered in robotic applications.

A recent comparison [75] shows that there is no single approach that is always outperforming the other approaches, even taking into consideration a single metric, however, SE-Sync shows superior performance both in terms of execution time and accuracy of the solution in many cases [75], showing how it could be the go-to solution for future SLAM back-ends.

3.1.2 Feature based

The two most important early works in feature-based visual SLAM can be identified in MonoSLAM [33] and Parallel Tracking and Mapping for Small AR Workspaces (PTAM) [82].

MonoSLAM is one of the first successful examples of monocular SLAM, as it achieved drift-free performance which was before inaccessible from Structure from Motion techniques (SfM). In MonoSLAM the strong correlation in the camera measurements is taken into account, using at the time popular EKF-based approach, where the state of each tracked 3D point was part of the state of the Kalman filter and so for each element of the map both the state and the Gaussian uncertainty were estimated:

$$\hat{x} = \begin{pmatrix} \hat{c} \\ \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix}, P = \begin{bmatrix} P_{cc} & P_{cy_1} & P_{cy_2} & \dots & P_{cy_n} \\ P_{y_1c} & P_{y_1y_1} & P_{y_1y_2} & \dots & P_{y_1y_n} \\ P_{y_2c} & P_{y_2y_2} & P_{y_2y_2} & \dots & P_{y_2y_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{y_nc} & P_{y_ny_2} & P_{y_ny_2} & \dots & P_{y_ny_n} \end{bmatrix}. \tag{3.32}$$

This is the state used by MonoSLAM (and many other full-state EKF-based visual SLAM approaches), where the $\hat{x}$ notation represents an estimate for the variable x, $\hat{c}$ represents the state of the camera, position and attitude, while y_n represent the n feature which is present in the state. For modern standards, the map looks extremely sparse, due to $O(n^2)$ computational complexity of the Kalman filter and the limitation of the hardware at that time. They keep in the map only 100 map points, also initialization had to be performed on an object of known size.

PTAM introduced the concepts of separating tracking and mapping from a computational point of view: in PTAM one thread has the task to create map points, and another thread has the task to localize the camera with respect to the map points which are identified in the current frame. Abandoning the full-state EKF approach and with the use of keyframes, PTAM was able to produce and maintain a much larger map compared to MonoSLAM. This time, initialization did not require an object of known geometry, but instead, it required user cooperation for the identification of the first 2 keyframes.

The most common form of Visual SLAM is composed of a front-end, a back-end, and a pose-graph optimization, and ORB-SLAM [113] has been one of the pioneers of this framework. ORB-SLAM begins to build from PTAM, addressing many of the most significant PTAM limitations. ORB-SLAM introduces a fully automatic initialization, which does not require any prior knowledge about the

scene by calculating at the same time a homography and a fundamental matrix and choosing one model or another by the mean of the score, based on symmetric transfer error [78].

The logic behind this strategy is that in planar scenes the ego-motion of the camera can be well explained by homographies while estimating a fundamental matrix could yield to an incorrect motion reconstruction, due to the twofold planar ambiguity [96]. ORB-SLAM extends the intuition of PTAM which splits in different threads mapping and tracking and adds another thread, which is in charge of loop closing.

The tracking thread performs also local BA, a key step to enforce the survival of the fittest for both keypoints and keyframes, enhancing at the same time robustness, the quality of the map, as well as minimizing the local drift that occurs tracking the camera position frame after frame. ORB-SLAM builds on several other concepts and methods, such as a pose-pose covisibility for bounding the computational efforts [148] [104], a loop closing strategy that is scale-aware (Strasdat et. al [147]), a bag-of-word approach called Distributed Bag-of-Words (DBoW2) [54] for place recognition. ORB-SLAM builds a *covisibility graph* and an *essential graph*.

The covisibility graph is an undirected weighted graph that represents the relationship between keyframes, with the weight θ on each arc being the number of common observations. Covisibility information is useful for triangulating new keypoints, where the newly created keyframe is used in conjunction with frames directly connected to it in the covisibility graph. The essential graph instead represents a subset of the covisibility graph, which contains all the nodes, but not all the edges. The essential graph is built using a spanning tree, starting from the first keyframe, plus edges with $\theta > 100$ and loop closure edges. The essential graph is a crucial step to improve optimization efficiency, indeed when performing pose graph optimization yields almost the same results as a map-wise motion and pose bundle adjustment.

Places are recognized by the use of DBoW2. The visual vocabulary is created offline, using ORB descriptors, on a large database of various images. An inverted index is built which relates to which keyframe each word is found. Specifically, in ORB-SLAM overlapping keyframes from a visual word's point of view are grouped using the covisibility graph. In order to speed-up feature matching, loop detection, and relocalization, DBoW2 is used also to retrieve candidates for keypoint descriptors brute-force matching. ORB-SLAM was further developed successfully into ORB-SLAM 2 [111] and ORB-SLAM 3 [25]. ORB-SLAM 2 extends the previous work to allow the use of stereo and RGB-D systems and introduces a localization-only mode. In ORB-SLAM 2 a full BA is also performed in a separate thread after the pose graph optimization on the essential graph, such full BA is performed online, while all the other SLAM operations are running.

Stereo keypoints are classified as *close* if their relative depth is less than 40 times the baseline, or as *far* otherwise. Close keypoints are immediately triangulated and used for mapping, while far keypoints can provide accurate rotation information and might be triangulated using more than two views.

ORB-SLAM 3 does bring a visual-inertial integration that relies on *maximum a posteriori* (MAP) estimation, a multi-map system, called the *Atlas* system [41],

that can allow merging maps created when the tracking was lost, an improved-recall place recognition, still based on DBoW2 and an abstract camera representation.

Visual-inertial combination allows VSLAM to add robustness in presence of surfaces with flat gradients, various blurs, and occlusions, and, in the case of monocular visual SLAM, it adds precious scale information. The visual-inertial approach in ORB-SLAM 3 builds on ORB-SLAM-VI [112], it expands the available camera models and speeds up initialization. Initialization is achieved at a much faster rate taking into consideration three key observations:

- Monocular SLAM is already able to produce very accurate maps, only with unknown scale [113]. Having a map can help in IMU initialization.
- Scale estimation converges faster when it is represented as a variable in an optimization problem [147], compare to estimating it in BA.
- IMU uncertainties have to be known *apriori* in order to avoid large and unpredictable errors [24].

Following these observations, first, a map is created without the help of IMU sensor data, shortly after the IMU-related variables are estimated using MAP estimation, taking into consideration a stack of sensor readings obtained through the initial keyframes, finally, a joint optimization is performed to refine together both visual and inertial parameters.

The Atlas system comes into play as soon as tracking is lost, instead of performing only attempts to relocalization, attempts to re-initialize are made. In case a re-initialization is achieved, a new map is created, now called the *active map*. The active map is disconnected from another(other) map(s) until a match is found in the Atlas database, with the help of DBoW2 and various geometric and inertial verification. When a match between different maps is identified and verified, map merging starts.

Due to the fact that the map merge process could take a lot of time, due to many overlapping elements, this process is divided into two different steps. First, a so-called *welding window* is defined and an initial merge happens here, by utilizing the now neighbor keyframes between the active map and the non-active map that is subject to the fusion, second, the correction previously found is utilized to merge the rest of the two maps, by the mean of pose-graph optimization.

The place recognition system still is based on DBoW2 but compared to the method present in ORB-SLAM [113], the order of consistency checks is inverted: the candidate keyframes are first checked for geometrical consistency and then for local consistency, this inversion directly boosts recall.

ORB-SLAM 3 makes the system compatible with any camera model, as the ePnP algorithm used for relocation has been substituted with a maximum likelihood PnP algorithm [155].

3.1.3 Direct methods

Direct methods oppose feature-based methods by the fact that they are able to use all the information present in the image, while feature-based methods are able to exploit only an abstraction of the image, in general in the form of corner information.

Human-made environments contains a lot of straight and curved edges which are not taken into account by corner-centered feature-based methods.

The first direct approach-based SLAM is the so-called Large-Scale Direct monocular SLAM (LSD-SLAM) [42], which is now presented to the reader.

A minimal representation of the camera pose is defined in $\xi \in se(3)$, from frame i to frame j it is written as ξ_{ji}. Two images are aligned by minimizing through Gauss-Newton the following energy function:

$$E(\xi) = \sum_i (I_{ref}(p_i) - I(\omega(p_i, D_{ref}(p_i), \xi)))^2. \tag{3.33}$$

Where $I_{ref}(p_i)$ represents the intensity of the image I_{ref} at point p_i, ω a projective warp function (see [147]), $D_{ref}(p_i)$ represents the depth D_{ref}, relative to image I_{ref}, at point p_i, and in general $(I(\cdot))$. Minimizing $E(\xi)$ equals performing a maximum likelihood estimation for ξ, assuming that all the residuals are independently and identically Gaussian distributed. During each optimization iteration, a weight matrix is computed to down-weight all the large residuals.

Initialization happens with the insertion of a keyframe, with a random depth map with large variance, then the system automatically, after the insertion of a series of other keyframes, should converge to the true depth (up to scale). The map is represented as a pose graph of keyframes, the inverse depth map, and its variance. The depth map becomes defined only in regions of an intense gradient. Contrary to ORB-SLAM, the tracked frames are utilized to refine the depth map by performing small-baseline stereo comparisons, and potentially also adding new points to the map. As a new keyframe is inserted into the system, a set of ten candidate keyframes is picked up with the help of OpenFABMAP [60], in order to evaluate a potential loop closure. To avoid closing a wrong loop, a reciprocal tracking check is performed. For each loop closure candidate keyframe $\mathcal{K}_{j_k}$, it is then estimated $\xi_{j_k i}$ and $\xi_{i j_k}$, and only if the two estimates are statistically similar the loop closures occur. The map is continually optimized in the background using pose graph optimization, with the help of the framework g2o [64].

Another popular direct method is Direct Sparse Odometry (DSO) [44], which was first published as just visual odometry, and then extended to SLAM with the addition of loop detection and closure [56]. DSO is a sparse and direct monocular visual odometry that jointly optimizes all the model parameters: the poses, the camera intrinsic, and the inverse depth values. DSO takes into account photometric calibration, lens attenuation, gamma correction, and known exposure time. DSO utilized a specific image formation model [43] in order to account for non-linear intensity response function and vignetting, the first operation that is performed is the correction for these two effects. A photometric error of a point is defined, as the sum of squared differences, over a set of neighbor pixels experimentally chosen, cleverly this neighbor is chosen in a way that enables Streaming SIMD Extensions (SSE) optimization. The photometric error p, observed in a frame j is so defined:

$$E_{pj} \coloneqq \sum_{p \in \mathbb{N}_p} w_p \left\| (I_j[p'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[p] - b_i) \right\|_\gamma \tag{3.34}$$

Where $\mathbb{N}_p$ represents the set of neighbor pixels, t_i and t_j the exposure times of the images I_i, I_j, $\|\cdot\|_\gamma$ represents the Huber norm and p' represents the projected point position of p. The gradient-dependent weighting w_p, with the goal of down-weights pixels with a high gradient, is so defined:

$$w_p \coloneqq \frac{c^2}{c^2 + \|\nabla I_i(p)\|_2^{\,2}} \tag{3.35}$$

Where c represents the camera intrinsics and $\|\nabla I_i(p)\|_2^{\,2}$ represents the norm 2 squared of the point p gradient.

The full photometric error over all the frames and points is given by:

$$E_{photo} \coloneqq \sum_{i \in \mathcal{F}} \sum_{p \in \mathcal{P}_i} \sum_{j \in obs(p)} E_{pj} \tag{3.36}$$

Where i represents all the frames $\mathcal{F}$, p over all the points in the frame i and $obs(p)$ in which the point p is visible.

The tracking is performed by a front-end which determines the error terms in a local E_{photo}, composed of 7 frames. The front end also provides initialization for the parameters needed to optimize E_{photo} and decides when a point or a frame should be marginalized.

New keyframes are inserted by taking note of the mean square optical flow from the last keyframe. New keyframes are also inserted in case the camera exposure time changes significantly.

In their papers, the authors show that DSO, compared to LSD-SLAM, greatly outperforms in terms of accuracy in sequences where no loop closures can be exploited by LSD-SLAM, and can produce a similar semi-dense map.

A version of DSO includes loop closure called LDSO [56]. LSDO favors corner-like features in the front end of DSO, this is because corner-like features are repeatable and represents something easily recognizable in order to close a loop. Shi-Tomasi is used to select the corner features, which are added to the points detected by the standard DSO method. The corner descriptors for loop closures are calculated on points that are tracked by the front end so that a depth estimate exists. LDSO so utilizes a Bag of Word approach for closing the loop. Loop closure candidates are verified geometrically and in the similarity transformations 3D space $SIM(3)$ by the mean of geometric error estimation. LDSO shows comparable performance to ORB-SLAM 2 in most of the sequences in the KITTI dataset.

3.1.4 Hybrid methods

Hybrid methods are methods that actively use both direct and indirect features. Direct and indirect features are not at odds with each other, and so methods that are able to exploit both can benefit from the robustness and repeatability of the corners as well as the ability to operate on cornerless.

A notable work in this space is SVO [50]. SVO extract features only when new 3D points need to estimate. Feature matching is an implicit result of direct motion estimation. SVO uses hundred of small patches to roughly estimate the motion through the minimization of the *photometric error* and match features, then it proceeds to perform a three-step bundle adjustment, first a *motion-only* BA, then a *structure-only* and finally a local BA. The photometric error is calculated between pixels corresponding to the projected location of the same 3D points, this is expressed with the following equation:

$$T_{k,k-1} = \underset{\boldsymbol{T}}{\operatorname{argmin}} \int \int_{\bar{\mathbf{R}}} [\delta I(\boldsymbol{T}, \boldsymbol{u})] d\boldsymbol{u} \tag{3.37}$$

Where $T_{k,k-1}$ represents the transformation between two consecutive camera poses ($k-1$ and k), δI is an intensity residual defined by the photometric difference between pixels observing the same 3D point, $\bar{R}$ represents an image region for which the depth d_u is known in the previous image $k-1$ and for which the back-projected points are visible in the image k.

Similar to ORB-SLAM, SVO utilizes a thread for camera motion and a second one for mapping, in order to maintain responsiveness to new incoming frames. Each 2D feature has a corresponding probabilistic depth filter which is initialized when a new keyframe is created. The depth is refined in a Bayesian fashion and when the uncertainty is low enough, a point is added to the map. SVO has been extended to exploit also line segments [62], enhancing robustness in structured environments. SVO has been improved by the original authors with SVO 2.0 [51], bringing support to multi-camera systems and with SVO Pro [122], which brought support for visual-inertial SLAM with loop closure using a similar approach to the one present in ORB-SLAM (DBoW2 and pose-graph optimization).

3.1.5 Deep-learning based

Artificial intelligence has started to dominate the field of computer vision, enabling previously impossible applications thanks to their "hands-on" prior knowledge captured by artificial neural network weights.

Learning to estimate motion and map enables VO and VSLAM algorithms to exploit prior knowledge about the environment to produce estimates impossible before, like producing depth maps with a single monocular frame [141] [128] [53] [106].

The need for a supervised algorithm that needs per pixel depth in order to be trained has been overcome with a self-supervised algorithm with the help of proper loss functions and image formation models [61].

Networks like the DeMoN (Depth and Motion Network for Learning Monocular Stereo) network [154], show that specialized networks can be built to estimate the ego-motion between consecutive camera poses, together with a depth map, all by needing only two image pairs as input, so regardless of intrinsic camera parameters.

The DeMoN network is constituted of three different works: a bootstrap network, an iterative network, and a refinement network.

The bootstrap network is specialized in creating a first estimate of the depth map and the motion estimation and is built using a first encoder-decoder which

computers the optical flow and a confidence map, second another encoder-decoder takes as input the optical flow, its confidence, the two image pairs and the second image warped with the estimated optical flow. Its output is the depth map, surface normals, and camera motion.

The iterative networks have the same architecture as the bootstrap network, but it takes additional inputs, specifically it converts the depth map and the camera motion into an estimated optical flow field, and it feeds itself three times.

The refinement network upscales the depth estimated at the third cycle of the iterative network to the input image size. The representation of the network can be found in Fig. 3.1.

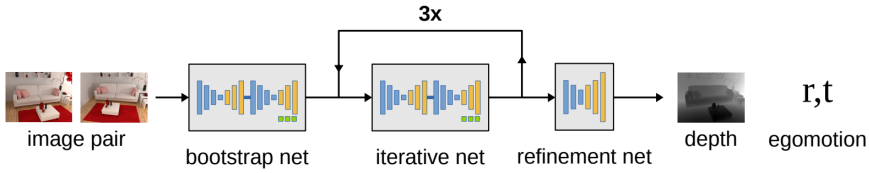

Figure 3.1: Overview of the DeMoN Network. The networks take as input two image pairs and estimate depth and ego-motion, without being aware of lens distortion, camera intrinsic, and photometric calibration. *Image from article [154].*

Training of the network is performed in a supervised fashion with three different losses: a point-wise loss which includes inverse depth, surface normals, optical flow and optical flow confidence, a set of motion losses, and a scale-invariant gradient loss which is regularized in order to penalize relative depth error between neighbor pixels.

A naïve VO system could be built just by concatenating estimates from the DeMoN network.

Deep learning networks developed specifically for VO have been developed, showing incredible performances.

An example is Deep VO [157], where the authors have utilized AlexNet [84] for performing transfer learning: AlexNet was modified to take as input two sequential images, with the object to regress on the relative position and relative attitude. Deep VO is able to perform well only in environments known at training time [84].

A more recent approach called UnDeepVO [92] exploits stereo images in order to perform unsupervised learning, at inference time instead the network requires only consecutive monocular images. UnDeepVO is able to estimate both the differential pose, as well as the depth map, similar to the DeMoN network. The performance in terms of accuracy, measured on the KITTI dataset, is able to outperform ORB-SLAM (when running as VO by disabling the local mapping, defined in the paper as "ORB-SLAM-M"), As a personal note on UnDeepVO, the fact that they disabled local mapping in ORB-SLAM would not allow ORB-SLAM to function even as only a VO, I expect them to have disabled the loop closure detection, not the local mapping.

DeepSLAM [93] is a very recent approach that combines unsupervised deep learning with pose-graph optimization, in order to achieve an almost solely deep

learning-based monocular visual SLAM. Similar to UnDeepVO [92], at training time stereo images are supplied, while at inference time only sequential monocular cameras are supplied, making DeepSLAM a monocular VSLAM system. DeepSLAM utilizes three different networks, denominated in the paper with the suffix "-Net": a Mapping-Net, a Tracking-Net, and a Loop-Net. The Mapping-Net is a convolutional autoencoder that has the goal to produce a 3D estimate of the environment. The Tracking-Net is a convolutional recurrent NN that estimates the camera ego-motion and Loop-Net is a CNN (Inception ResNet v2 [149]) which maps images into a feature vector for loop closure detection. Loops are found when the cosine distance between the current image feature vector and one of the images in the database is below a certain threshold. Loop closure is then completed with a pose-graph optimization. DeepSLAM is evaluated on the KITTI [59] dataset where the performance, measured as a percentage deviation from GT on the path from 100m to 800m, is considerably worse than ORB-SLAM, but much better than SfMLearner [171]. DeepSLAM is also evaluated on the RobotCar dataset [101]: the sequences in the RobotCar dataset contain atmospherically phenomena like rain, and images captured at night or on a day with a strong sun, in these scenarios DeepSLAM becomes the only system being able to maintain tracking during these sequences, showing how a data-driven approach can deeply improve robustness in such scenarios.

LIFT-SLAM [167] (Learned Invariant Feature Transform) [21] is another deep learning-based visual SLAM, but to a much less extent when compared to DeepSLAM [93]. The authors of LIFT-SLAM recognize that no deep learning-based SLAM is able to outperform geometry-based methods, but the deep learning basedmethods are able to keep the VSLAM when the images are rich in noise. LIFT-SLAM systems, instead of relying only on deep learning methods for estimating motion, combine it with traditional geometry-based VSLAM. More in detail, the so-called "LIFT" network [167] is a deep network that performs feature extraction, the rest of the system is based on ORB-SLAM. The LIFT network takes as input image patches and produces in the output the potential location of the feature point, then it estimates the orientation of a patch around the potential feature point, and lastly, a feature vector is produced. Compared to the original ORB-SLAM system, the creators do not run tracking and mapping threads at the same time, but they are sequential.

As a personal note, this already could explain superior performance in terms of position and attitude errors, due to the fact that if frames come faster than the ability of the mapping thread to generate points, the tracking could be lost or the performance could be sub-optimal, due to a smaller amount of map points being available to estimate the ego-motion. LIFT-SLAM has been tested on the KITTI and on the Euroc dataset [23] and shows slightly better performance compared to ORB-SLAM, showing why it could be beneficial to combine learning features with one of the best-performing geometry-based VSLAM still today.

3.2 Underwater SLAM

Performing SLAM underwater has been mostly performed using acoustic sensors [164]. The reason for such a choice has been already explored in this thesis, but to summarize it here we can say that water (especially salt water) greatly attenuates electromagnetic waves. Nevertheless, approaches using cameras have been explored, which are able to provide color information on the map and enable small and/or low-cost robots to perform localization and mapping underwater.

3.2.1 Acoustic-based Underwater SLAM

Acoustic-based or aided navigation for AUVs is by far the most popular option, given that sound travels in the water much more easily than electromagnetic waves. Acoustic measurements are also essentially not influenced by water turbidity.

Sensors

There are several acoustic-based sensors that could be used to aid produce SLAM estimates, the first kind those are acoustic Doppler current profiler (ADCP), which for underwater vehicles is used as Doppler-Velocity Log (DVL). DVL emits acoustic pulses in different directions and by sampling the acoustic pulses received back and measuring the Doppler shift resulting from the fact that the AUV is moving and the seabed is not, a velocity vector can be estimated as

$$f_o = \frac{v_{sound} + v_o}{v_{sound} + v_e} f_e. \tag{3.38}$$

Where v_{sound} represent the speed of sound in a particular medium (water in this case),v_o and f_o are the velocity and frequency of the observer, as the observer is the seabed, in this case, v_o is 0 and f_o represents the frequency of waves reflected, and v_e and f_e are the velocity and frequency of the emitter, in this case of an underwater vehicle. The frequency of the emitter is known, and so the only variable in this equation is v_e, the vehicle velocity. DVLs are able to directly observe the velocity of the underwater vehicle with a zero-mean bias and so are often modeled as zero-mean Gaussian white noise [40].

Being characterized by a zero-mean noise makes DVLs great sensors for long underwater operations in unstructured environments. DVLs are not always able to provide estimates, this can happen when the so-called *bottom-lock* is lost. First of all, DVLs have operating ranges, the underwater vehicle could be located outside of this range (too close or too far from the seabed). Other characteristics which could make DVLs lose the bottom-lock are characteristic of the seabed, for example, the seabed could be sloped or could be composed of a material that easily attenuates the acoustic waves that DVLs emit. Raw DVLs measurements can also be loosely-coupled with INS measurements [150].

Side-scan sound navigation and ranging (sonar) are other different kinds of acoustic sensors, they are usually composed of an array of transducers, and they

are located on the side of a boat or underwater vehicle, or located on a device that is towed by the boat/underwater vehicle.

Side-scan sonars are able to directly recover underwater 3D structures by measuring distances through acoustic echos, mathematically the distance measured follows the so-called active sonar equation:

$$SnR = Sl - 2T_l + T_s - (n - A_g) \tag{3.39}$$

SnR stands for signal-to-noise level (expressed in decibels), S_l stands for signal level, T_l transmission level, T_s target strength (a factor dependant on the reflecting object and its shape), n noise level in the receiver, A_g array gain, given the use of multiple sensors to receive the same signal (to counter-act the receiver noise).

Sector-scan sonars are a kind of sonar that produces 2-dimensional images by performing a sweep up to 360 degrees and they are mostly used for obstacle avoidance [11].

Multi-beam echo-sounder (MBES) is another sonar technology for recovering the 3D shape of the seabed, compared to Side-scan sonar, MBES uses beamforming (directional signal transmission and/or reception) and works by emitting a fan-shaped array of acoustic waves directly under the transceiver.

Synthetic aperture sonar (SAS) are amongst the most advanced kind of sonar, that are capable of producing very high precision maps (down to the cm) for hundred of meters [38], unfortunately, SAS is also the most expensive kind of sensors. In Fig. 3.2 a comparison between sector-scan sonars, side-scan sonars, and synthetic aperture sonars are shown.

3.2.2 Acoustic-based SLAM

An example of a side-scan sonar SLAM is supplied by ATLAS ELEKTRONIK [145], which shows a simple and effective features-based approach. The scan is firstly down-scaled and then features are identified with the help of a threshold, these features are then inserted into a map, and features coming from new scans are then associated with the help of *joint compatibility branch and bound* [116] and heuristics like ambiguity check and time gating, a strategy that addresses specific artifacts of side scan sonar imaging. The state estimation is based on an EKF.

SAS systems achieve higher spatial resolution by utilizing sensor multiple sensor readings, obtained at different times, and this is achieved in combination with very high-quality INS systems. The authors of [28] have proposed to use SLAM instead for performing a fusion of acoustic and navigational data, abandoning the Gaussian framework for a factor graph formulation.

3.2.3 Underwater Visual SLAM

The most popular approach for performing underwater visual SLAM has been the use of stereo camera rigs [127] [71] [161] [27].

A classic approach involves the use of an EKF for performing sensor fusion and state estimation [127]. Features estimated through SURF are transformed to 3D

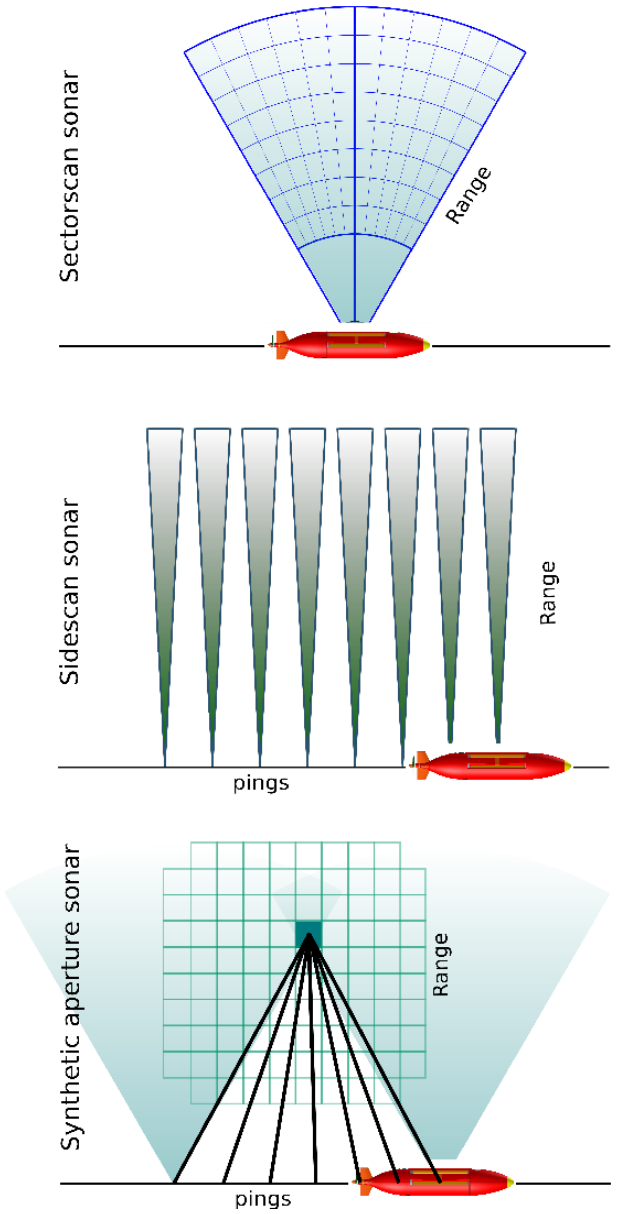

Figure 3.2: In this figure the difference between Sector-scan Sonar, Side-scan Sonar, and Synthetic Aperture Sonar are illustrated. *Image from article [66].*

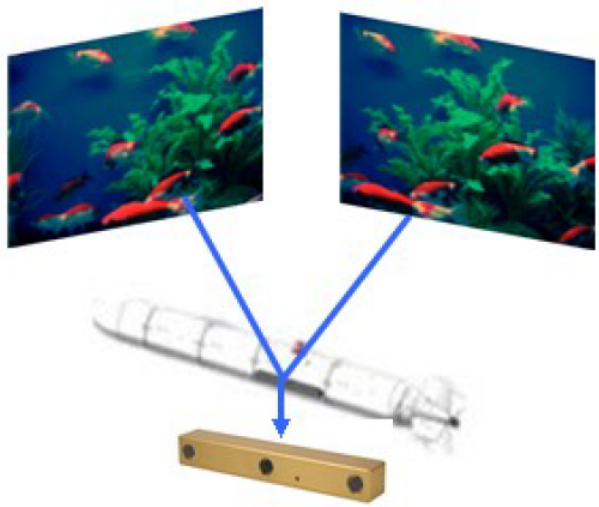

Figure 3.3: Stereo camera location in [127]

points using the stereo camera equation and are incorporated into the EKF SLAM state. The stereo camera is placed similarly to a side-scan sonar, see Fig. 3.3.

In the work of Weidner N. et al. [161] a strategy that specifically targets the exploration of underwater caves is developed. Underwater caves receive a very small quantity or no natural light, so visual SLAM has to rely only on artificial light. In their work, they had multiple independent artificial lights, one coming from a light source dedicated to the stereo camera rig, and multiple sources coming from the divers which accompanied the stereo camera rig. The light sources coming from the divers make traditional visual odometry almost impossible, with ORB-SLAM being found to be the most successful approach.

The authors utilize adaptive thresholding to identify areas with different illumination, then a canny edge detector is used to identify those boundaries, then for every point of this contour, a SURF descriptor is calculated in order to perform feature matching between the two stereo images and to perform triangulation. On a critical node, no information about loop closure is present in the paper, likely because features taken on the edge between a sharply illuminated area and a dark area are substantially impossible to replicate.

Carrasco et al. [27] propose a stereo graph-based SLAM with features kept out of the graph. Each graph node represents a pose and loop closure is constrained at a region of interest (de-facto making loop closure capabilities sensible to drift, as the region of interest, could be too small to enclosure the current loop closure opportunity). Similar to ORB-SLAM and many other SLAM methods, the framework g^2o is used for performing graph management and optimization.

Given the fact that not always visual information is available due to lack of illumination, navigation path, and seafloor structure, a common strategy is to fuse

visual information with INS, both for monocular systems and stereo systems [105]. The authors [105] propose a new camera model which is reported to achieve a 10% lower reprojection error than the pinhole camera model. Projection errors and photometric errors are jointly optimized and features are matched with the help of a local optical flow, specifically designed to improve robustness in keypoint tracking. The authors have tested their work on the AQUALOC dataset [49], as well as a new dataset that they have built in an artificial environment where they could obtain a ground truth accurate to the millimeter. The results are excellent in the artificial environment, but less impressive in the natural environment presented by [49].

Deep learning-based VO and SLAM have not been popular in underwater-specific visual ego-motion and mapping applications, simply because the most effective algorithm in this domain are supervised algorithms, and there has no public dataset available with, for example, a 3D ground truth coming from a laser sensor. However, Teixeira et al. [152], have utilized the database recovered by the UNEXMIN UX-1 [103], which provides images from flooded deep mines, with ground truth pose data composed by a sensor fusion from multiple sensors on the robot, like IMU, DVL.

The authors developed a new network that takes as input both GeoNet and SfmLearner predictions and IMU readings (not needed at inference time), then proceeds to stack several Long short-term memory (LSTM) units. The loss function is a simple MSE, with the rotation parametrized as quaternions, present in the loss function as a quaternion difference, enclosed in the L1 norm.

According to the authors, in the two sequences analyzed, traditional methods like ORB-SLAM 2 and LDSO were not able to track at least two third of the sequence, while SfMLearner, GeoNet, as well the network proposed by the authors [152] were able to track a higher amount of these sequences, with the network proposed by the authors outperforming the two other networks.

Loop closures remain an absolutely crucial part of visual SLAM systems, as a wrong loop closure could render the SLAM estimates no longer useful. A dedicated deep-learning image descriptor for loop closure called NetHALOC [18] demonstrates how a simple and fast CNN can perform dimensionality reduction for underwater images (similarly to [87]), which is fundamental for fast image comparison, and so loop closure.

NetHALOC has an encoder-decoder structure, where the encoder section is composed of a series of convolutional and max pooling layers, while the decoder is instead composed of a flattening layer followed by three dense layers. The output of the decoder is not an image, it is instead a so-called Global Image Descriptor (GID). The name NetHALOC represents the network version of HALOC [115], a GID which has shown to be quite effective underwater [114], for this reason, the GID learned by NetHALOC has the same dimensionality (384) of the HALOC GID. The NetHALOC network is cleverly trained using an input of an image, and as output, the HALOC GID of an image closes the loop, in this way the network is trained to produce the GID of an image that could close the loop.

The network could be trained in a supervised way, or in an unsupervised way. The unsupervised way is based on the generation of another image through random rotation, shifting, and scaling of the input image, and so the target is the

GID newly generated image. Results are mixed, with the Aqualoc dataset, having a recall of only 66%, the authors themselves write that NetHALOC does not represent a real breakthrough, but still represents a very interesting attempt that can be further refined.

3.2.4 Future trends: a short glimpse

State-of-the-art in the presented research fields has moved forward very rapidly in the last years and the field of underwater vision and underwater visual SLAM has progressed a lot, an example of future trend can be seen in the use of variational autoencoders for place recognition [158] and unsupervised monocular motion estimation based on view synthesis and photometric loss [29].

Another field where a lot of progress has been made is obstacle avoidance, a trend is to use multi-agent reinforcement learning, and adversarial networks [47] in order to perform obstacle avoidance with single or multiple robots at the same time.

It has to be noted that the scarcity of underwater visual datasets has been always a problem for developing robust underwater visual SLAM systems, and indeed was a problem for the Candidate, however, this issue might be a problem of the past, thanks to synthetically generated datasets [109, 173].

Chapter 4

Contributions to Underwater Visual SLAM

This chapter will present an overview of the research performed during the period spent at NTNU. The research started with a series of field experiments in order to gain experience and knowledge in operating vision systems underwater, it continued exploring applications of deep artificial neural networks to underwater VSLAM and has been concluded with the identification and development of a set of effective upgrades to ORB-SLAM 2 in order to generate a robust underwater visual SLAM system.

4.1 Obstacle Avoidance exploiting Stereo Vision

The research performed on stereo imaging and obstacle avoidance provides empirical evidence of stereo-camera-only underwater obstacle avoidance.

The research was performed through a series of field experiments, utilizing an ROV equipped with two industrial cameras and an active illumination system. The hardware setup is reported in detail in Paper A: Stereo obstacle avoidance.

The test location was a shipwreck in the fjord of Trondheim (Norway). Camera calibration has been performed in water utilizing the pinhole camera model, exploiting three different toolboxes: OpenCV, MATLAB, and Caltech. The calibration results have been compared, with the MATLAB-integrated stereo calibration toolbox yielding the best result in terms of reprojection error.

Three different disparity map generation strategies have been tested: Sum of Absolute Difference (SAD) which is a local strategy, a semi-global strategy called semi-global block matching [72] and a global strategy, the Kolmogorov and Zabih stereo matching algorithm [83]. SAD with a 55x55 window (approximately 21/14 times smaller than the rectified image) performed the best in terms of the ability to detect obstacles in the disparity map. In addition, various pre-processing techniques have been tested in order to improve the disparity map, but they didn't provide any sensible improvement.

Once obtained the disparity map and 3D points, there comes a need for post-processing in order to obtain estimates useful for obstacle detection and navigation. Several approaches have been tested to filter out 3D points irrelevant to obstacle detection and we have found that a modified version of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (a density-based clustering that classifies data points in low-density area as noise, requires only one parameter ϵ, which defines the circular search radius) performed the best. The modification involved dynamically updating the ϵ parameter based on the distance of the 3D point from the camera. The reason why this modification fits very well with stereo-camera point cloud clustering and filtering is that the points in a point cloud generated through a stereo-camera system are sparser, as they are further away from the camera.

Utilizing the same data acquisition for visual obstacle avoidance, rudimentary visual-inertial odometry based on SURF features and the Nelder-Mead simplex method [117] optimization has been tested and has been quite successful. During the experiments, a so-called *autonomy layer* of the ROV control system took care of activating a pre-determined change of course as soon as the visual system would identify an obstacle.

4.2 Machine Learning Elements for increasing SLAM Robustness

One of the main reasons for the under-performance of visual motion estimation techniques is the inability to cope with the overall lower quality of underwater images and the potentially high amount of moving elements in the scene (see Chapter 2, Section 2.4).

4.2.1 Loop closure candidate retrieval

The research performed on loop closure candidate retrieval contributes to the field of underwater visual SLAM by demonstrating how an autoencoder image-matching procedure can substitute loop closure candidate retrieval based on the bag-of-words approach.

Loop closures are a crucial part of SLAM systems that allow bounding the estimation error, which could be significant if the SLAM estimate is extended in time and does not benefit from any global positioning system.

Closing a loop requires recognizing a scene or image seen before, this then requires a database where information is stored about each scene or image previously seen and rises the problem of the limitation of computational resources. It indeed requires both a very compact image or scene representation and a fast way to search in such a database for possible matches.

A solution to this problem was to represent images as *bag-of-words* vocabulary and build an inverted index [8] [110], unfortunately, this method requires the construction of a vocabulary that will be dependent on the kind of images supplied

during the training phase, also utilizing only information derived from features does not guarantee that the images do belong to the same scene.

The solution proposed in Paper B Paper B: Loop closure detection is to avoid features to compare images, but instead perform a direct comparison, utilizing a highly compressed version of the original image, generated from the latest layer of a CNN encoder. The result is an image descriptor of a size of 4096 elements which is directly generated from the entire image. Cosine distance is then used to compare the current image representation with the previously collected ones through the cosine distance. To keep the computation time low and stable, the search for similar images is performed on the GPU. Utilizing an Nvidia TITAN 6Gb the search time has been demonstrated to be irrelevant to the number of entries for 4000 images (see Fig. 1 in Paper B: Loop closure detection).

Only after similar images are found, then a keypoint-based comparison is performed to validate the match, this ensures that the keypoint which are compared between each other are coming from images that have to look similar. Comparisons with FAB-MAP 2.0 [30] using underwater and street datasets have shown (see Table 1 in Paper B: Loop closure detection) that the approach performs retrieves loop closure candidates which a much higher precision, with a comparable run-time.

4.2.2 Improving feature tracking for visual ego-motion estimation

The research performed on improving feature tracking for visual ego-motion estimation contributes to the field of underwater visual SLAM by demonstrating how a simple convolutional neural network can be used to reject keypoints belonging to surfaces not suited for visual ego-motion estimation, and so improve feature-based visual ego-motion estimation systems.

The underwater environment can be a highly dynamic environment, especially near the sea bottom (a reference to many underwater visual ego-motion estimations) as many forms of life of all shapes and sizes populate the sea. The vast majority of visual ego-motion estimations estimate the ego-motion assuming that the environment is static, such assumption is often wrong in natural underwater environments: *school of fishes* and currents moving together large sections of vegetation are common occurrences and can create scenes that have a *coherent motion* that can be mistaken for ego-motion [90]. Considering also that many visual ego-motion estimation strategies are based on features. Filtering such features would increase the robustness of the ego-motion estimation.

An example of such a system, called *keypoint rejection system* has been developed as part of the research performed and presented in Paper C: Keypoint rejection system.

The strategy employed has been based on a *shallow* (a quite low amount of weights and layers) deep learning CNN trained to classify keypoints as *suitable* and *unsuitable* for visual ego-motion estimation. The information provided to the network for training and inference has been a square $65x65$ image patch (for images of 1280x1024) centered in a previously identified keypoint. A dataset of image patches that separates suitable and unsuitable keypoints for ego-motion estimation

was not publicly available, therefore, a method has been developed to manually build such a dataset.

The procedure involved visualizing an image with keypoints drawn on it and selecting an area of it through the mean of a series of mouse left clicks, with the goal of a complex polygon enclosing the highest possible amount of unsuitable keypoints. The procedure is then repeated until all the unsuitable keypoints are selected. Then all the keypoints present inside the polygons are used to generate the unsuitable keypoints patches, and the other keypoints are used to generate suitable keypoint patches instead. In this way, a very high amount of patches could be generated in a short amount of time.

The network utilized is a very simple CNN, composed of three layers of convolutional, max pool, and RELU concluded with a fully connected layer and a softmax function. Inference run-time performance of the network is put in a central spot: the analysis includes 32 vs 16 floating-point implementations through the Nvidia TensorRT library, batch inference, and single and multi-threaded implementation on both GPU and CPU.

The network prediction performance is reported in a form of a confusion matrix and tested against DynaSLAM [17] on a particular sequence which shows severe drift due to a moving scene, showing that this approach prevents initialization on several sequences where fishes move in front of the camera and produce a *coherent motion*. Inference time is also greatly reduced compared to the network present in DynaSLAM (Mask R-CNN).

4.3 Description of the Underwater Visual SLAM (UVS) System

The development of an improved monocular ORB-SLAM 2 for underwater applications contributes to the field of underwater visual SLAM by providing the research community with a new visual SLAM system to compare with, as well as providing hints on what could improve underwater performance in future underwater visual SLAM systems.

There was a desire to develop an underwater visual SLAM system "from scratch", utilizing and improving on several visual SLAM components observed to work well underwater.

Unfortunately, due to time constraints, the research focused on improving monocular ORB-SLAM 2 to achieve superior estimates underwater.

The are several reasons that lead to the decision to improve monocular ORB-SLAM 2.

In terms of the quality of the estimates and the robustness, a monocular camera visual SLAM is not desirable. There is indeed no doubt that stereo-vision systems can provide more accurate estimates than monocular systems. However, on small robots stereo systems can degenerate in a monocular system due to an insufficient baseline, the space might not be enough to fit two cameras and hardware malfunctions could affect one of the cameras of the stereo system. Having the possibility

of falling back to a monocular visual SLAM could be a critical feature to many underwater robotic applications.

ORB-SLAM 2 is a *feature-based* visual SLAM system. Several so-called *direct methods* have been proposed, such as LSD [42] and LDSO [56]. One of the potential benefits that direct methods would bring in underwater SLAM would be the ability to estimate the ego-motion in low light conditions and potential areas of the seabed lacking features.

The fundamental problem of such methods applied to underwater scenes is that the depth of a 3D point becomes a function of its intensity, which changes due to the non-linear absorption of light in the water.

Such an issue is accentuated by the fact that most likely the source of light illuminating the scene is rigidly attached to the robot, and therefore robot movements could translate into a change of illumination and so could produce an apparent change in the depth of the scene. To further complicate the matter, light absorption by water is also a function of salinity and temperature [126]. Even assuming that full control of all the physical variables involved could be achieved, such a solution would be impractical for many underwater applications.

LDSO has been tested underwater [88] and it has been shown to greatly underperform ORB-SLAM 2 in terms of Absolute Translation Error (ATE) and in terms of robustness, losing the ability to perform SLAM of many sequences.

An ideal solution would have been to develop a visual-inertial monocular SLAM dedicated for underwater, as IMU are likely to be present even in small robots and do complement very well with visual ego-motion estimation, as inertial sensors do not require any special environmental condition outside a robot to perform well. The reason why this path has not been followed relies on the fact that there are not many available underwater datasets that include inertial data. To further complicate things, it is most likely that many underwater visual-inertial attempts fail to provide better solutions than visual-only due to camera-IMU relative calibration being miss-aligned, as the camera-IMU calibration was performed in-air [65].

The research performed has identified many shortcomings of ORB-SLAM 2 for underwater SLAM and addressed them successfully.

The homography-fundamental matrix initialization in ORB-SLAM does not suit well underwater environments: a homography would work well only on planar surfaces, which is unlikely to exist in feature-rich underwater environments, and estimating the fundamental matrix for estimating the side-way motion with a 6-, 7- and 8- point algorithm is simply not recommended [119]. The idea is then to utilize the 5-point algorithm [119] instead and utilize 3 views instead of 2, this guarantees a unique solution (see Table 1. in [119]). As a conceptual note, all monocular visual ego-motion estimations which do not have access to external scale estimation, do benefit from initializing using three views, as monocular ego-motion estimation is structurally a three-view problem, due to relative scale estimation.

We identified a crucial performance limitation of ORB-SLAM: the way features are matched between frames. ORB-SLAM performs *brute-force* matching between those features that belong to the same vocabulary tree, to dramatically decrease the computational complexity. This procedure returns only a subset of all the possible and valid matches, as it is sensible to the performance of the pre-trained vocabulary.

Valid matches are the matches where the generated 3D point is validated through all the quality checks that ORB-SLAM performs on newly generated 3D points, like for example the epipolar constraint. Attempts made to re-train the vocabulary with underwater images didn't improve the overall performance. The solution is then to perform a real *brute-force* match, which goes through all the possible matches. However, a naïve implementation would not be fast enough, instead, because the descriptors that have to be matched are binary descriptors, a deeply optimized matcher [10], which is based on unrolled loops and the use of $X86\_64$ instruction *_popcnt64()* can be used. A speedup of 180X, compared to a naïve implementation, and 20% more valid descriptor matches can be achieved.

The conclusion is that the computational time required to match two sets of binary descriptors becomes just higher than the one achieved by the DBoW2-based brute force. Furthermore, as the matching involves two keyframes at a time, further parallelization can be achieved by launching a thread for each keyframe pair.

We introduced a fundamental architectural change: in ORB-SLAM the front-end and the back-end are running asynchronously. To allow ORB-SLAM to be as responsible as possible for new frames arriving, this is eliminated, in favor of synchronization between the front-end and back-end. The issue is that the front-end is indeed dependent on the back-end: in the case, frames arrive faster than the ability of the back end to produce map points, tracking will be lost, regardless of the quality of the scene. In terms of robustness, this is a clear design drawback, which can be easily addressed by forcing the front-end to wait for the back-end to have completed the map-point generation. In case of frames arrive while the front-end is waiting on the back-end, frames are inserted into a first-in first-out (FIFO) queue. Accumulation of frames in this queue is in general to be avoided (the consequence would be an always increasing time *delta* between the estimates and the current position of the robot/vehicle), and this can be achieved by correctly sizing the hardware required for a specific visual SLAM application.

We introduced a motion model to progress the camera attitude and position when the tracking is lost. This allows the immediate possibility of re-initialization, without losing the previously estimated poses and map and without having to wait for a loop closure to occur, as in ORB-SLAM.

Loop closure detection, which was utilizing the vocabulary for accelerating descriptor matches, is now utilizing the same full deeply optimized brute-force approach, increasing the number of closed loops by 4.5% in-air and 66% underwater.

In addition, we proposed a scale-agnostic station-keeping detection: the theory is that when performing station-keeping, global optimization could take place. Carefully considering the robot attitude derivative, the average angle between velocity vectors and the ratio of commonly observed map points is demonstrated using an in-air sequence 00 of the KITTI dataset, for the lack of better alternatives.

Lastly, the long-term ability to operate is addressed, with a specific pruning that limits the number of map points and keyframes. This not only guarantees the possibility to perform SLAM without any time limit but also bounds the computational complexity for loop closure correction and global BA. The pruning first addresses map points, creating a new entity in the SLAM system, the so-called *partially pruned keyframe*. By pruning map points first, the ability to close loops is preserved, as map points are pruned in a way that the remaining points are

uniformly distributed. Such strategy is also instrumental for path-planning and navigation, as pruning according to a uniform distribution, in absence of other information, maximizes the probability of maintaining surface reconstructability, given the Nyquist–Shannon sampling theorem. The creation of a partially pruned keyframe involves pruning 66% of the keypoints observed by such keyframes, to be able to close the loop of partially pruned keyframes, the loop detection threshold is lowered also by 66%. When all the keyframes outside the local map are partially pruned keyframes, full pruning of keyframes and relative observed map points is performed, starting from the keyframe further away from the active keyframe, and proceeds following the creation order through the essential graph.

The overall performance compared to ORB-SLAM is significant, with median RMSE being 21.5% better on in-air sequences and 213.85% better in underwater sequences, and in both in-air and underwater sequences, there is an increase in map points and loop closures. Public underwater sequences where ORB-SLAM was not able to complete the sequence, like RTMVO 5 and Aqualoc 01, can now be completed.

A private underwater sequence is also tested, called *Kjerringholmen north*, from the Norwegian location where it has been captured. The sequence is captured by an AUV of the Applied Underwater Robotics Laboratory (AURLab) [1] with a seabed-looking camera and a led light system. The AUV moves straight and it lowers itself until the seabed can be seen by the camera.

Calibration has not been performed on-site, but in a pool with a similar height from the seabed, see Fig. 4.1. ORB-SLAM on such sequence is not able to initialize, while LDSO produces completely wrong estimates, the images are very dark, so CLAHE is used to enhance them. With and without CLAHE, ORB-SLAM improved for underwater can initialize and provide estimates for all the sections of the sequence where it would be possible to expect a visual SLAM estimate, as outside of this section of the sequence the images are too dark, or simply the seabed is not in the visible.

ORB-SLAM 2 with all the robustness improvements here described has been called Underwater Visual SLAM (UVS), the paper describing this work is present in this thesis: Paper D: Underwater Visual SLAM system (UVS).

UVS has been made to run also on a small and low-power platform like the Nvidia Jetson Nano, see section UVS on Jetson Nano in chapter *Additional documents*, and on a series of sequences captured from Eelume's Snake robot docking, which include a sequence where the robot is performing underwater docking, a sequence where the camera looks at floating ice from under the water and a sequence where in front the camera appears first human-made structures and then the seabed. The document is present in this thesis Eelume Snake's Camera Sequences, in chapter *Additional documents*.

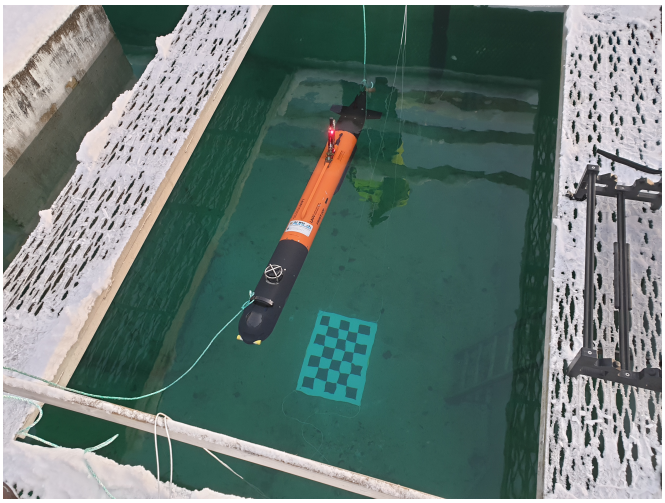

Figure 4.1: AURLab's AUV camera calibration, performed more or less at the same distance from the sequence *Kjerringholmen North* dataset [88]. Author: Marco Leonardi.

Chapter 5

Concluding Remarks

The research work presented in this thesis has explored several aspects of underwater vision, like stereo and obstacle avoidance, with a particular emphasis on improving monocular visual SLAM and its viability, robustness, and long-term operation. The performance of the research produced has been assessed with empirical full-scale experimentation as well as publicly available datasets and benchmarked against the *state-of-the-art*. The research questions that this thesis has aimed to answer are the following:

1. How should we design an underwater obstacle avoidance system exploiting stereo vision?
2. How could we increase the robustness of loop detection in underwater scenarios?
3. How could we increase the robustness of feature matching on underwater image sequences for visual ego-motion estimation?
4. What are the key elements necessary to address the limitations of ORB-SLAM systems when operating underwater?

5.1 Conclusion

Visual SLAM is one of the most complex tasks in computer vision, and sure trying to perform it underwater does not makes things easier. Water and the waterproof enclosures required for the cameras can deeply affect vision by introducing a series of non-linear effects.

Potentially most of these effects can be modeled and so addressed but could require additional sensors to sample the physical characteristic of the water, like salinity and temperature. Several works on the calibration of flat-port systems have been proposed [3] [99], however when it comes to empirical experiments which involve visual features for visual-ego motion estimation, such non-linear effects can be substantially ignored.

When it comes to calibration of a pin-hole camera model with a flat camera port, until the distance between the camera and the checkerboard is uniform and structures appear to be at a similar distance to the one used in calibration, the

3D reconstruction performance is not particularly affected, resulting in estimates precise enough for most purposes.

Visual ego-motion estimation performance and robustness in natural underwater environments can be achieved by utilizing feature-based methods. While direct, hybrid and deep-learning-based methods are not to be excluded. The further complexity present in underwater environments has proven that work has to be pursued to make them a viable choice. However, attempts of creating underwater visual odometry systems based on deep learning have been successful [152].

Robustness has been explored in several aspects, in particular loop closure, feature filtering, feature matching, and *dead reckoning*.

Robust underwater loop closure detection has been addressed by utilizing the compressed representation produced by a convolutional autoencoder and parallelizing the search for potential loops on a GPU. Utilizing deep learning methods for loop closure detection has been further researched by the community, both with unsupervised [22] and supervised [18] methods.

The presence of highly dynamic and unpredictable elements in the environment constitutes a real danger for visual-ego motion estimation systems underwater. *School of fishes*, uniformly moving elements of vegetation, or anything else which can be transported by currents, could represent a set of coherently moving features that has the potential to be mistaken for ego-motion estimation, filtering such features is a must for robust visual motion estimation and has been demonstrated to be able to be performed using square image patches centered around detected features and a small and efficient convolutional neural network.

The insufficient amount of feature points as well as their poor quality (for example, poor repeatability) have already been identified as problematic in underwater scenes [114]. For this reason, the departure from vocabulary-based matching is one of the key improvements for underwater scenarios. In ORB-SLAM 2 such departure has generated a great increase in performance in almost all the available metrics.

Monocular-only VSLAM requires the presence of a motion model able to perform *dead reckoning* and produce all the possible estimates, as soon as a sufficient amount of features can be identified in multiple frames.

The lifelong operation which may be necessary underwater requires careful memory management, which can be achieved by pruning the structures generated by the VSLAM. When pruning, it is important to maintain the possibility to perform loop closure. With such a goal in mind, map points can be pruned first, and the loop detection strategy updated to consider such pruning.

Station keeping is a typical operation in underwater robotics, in such settings, VSLAM components dedicated to mapping generation are not often needed, so the map and poses could be optimized. Detecting station keeping could be quite trivial with a stereo camera system but is indeed not with a monocular camera as scale information is missing. With a single camera, detecting station keeping can be achieved by carefully considering the ego-motion velocity vectors, the attitude derivatives, and the ratio of commonly observed map points in a *window* of keyframes.

5.2 Recommendations for further work

Feature and deep-learning-based hybrid visual SLAM systems are likely to outperform purely feature-based approaches in the near future, especially in robustness and ability to maintain the tracking, one of the most critical aspects when operating underwater, given the relatively low performance of in-air methods.

The way forward stands in creating a tightly coupled visual-inertial system that takes into account also the other sensors present on the robot, like the sonar system and the fully integrated SLAM system, which is able to robustly cope with the lack of one or multiple lacks of the sensors readings.

Preliminary experiments utilizing the Pinax model [99] performed during the research period, have shown the possibility to use the reprojection error in order to detect the level of salinity in the water, up to a single-digit precision. Utilizing such a strategy could help in achieving superior 3D reconstruction performance in presence of a camera model with salinity as a parameter, without the need for specialized sensors.

Chapter 6

Original Publications

6.1 Paper A: Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles

Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles

Marco Leonardi, Annette Stahl
Michele Gazzea
Department of Engineering Cybernetics,
Centre for Autonomous Marine
Operations and Systems, NTNU AMOS,
Trondheim, Norway.
Email: marco.leonardi@itk.ntnu.no,
annette.stahl@ntnu.no, micheg@stud.ntnu.no

Martin Ludvigsen
Ida Rist-Christensen
Department of Marine Technology,
Applied Underwater Robotics Lab,
NTNU, Trondheim, Norway.
Email: martin.ludvigsen@ntnu.no,
idaris@stud.ntnu.no

Stein M. Nornes
Department of Marine Technology,
Centre for Autonomous Marine
Operations and Systems, NTNU AMOS,
Trondheim, Norway.
Email: stein.nornes@ntnu.no

***Abstract*—Performing reliable underwater localization and maneuvering of Remotely Operated underwater Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) near nature protection areas, historical sites or other man-made structures is a difficult task. Traditionally, different sensing techniques are exploited with sonar being the most often used to extract depth information and to avoid obstacles. However, little has been published on complete control systems that utilize robotic vision for such underwater applications.**

This paper provides a proof of concept regarding a series of experiments investigating the use of stereo vision for underwater obstacle avoidance and position estimation. The test platform has been a ROV equipped with two industrial cameras and external light sources. Methods for underwater calibration, disparity map and 3D point cloud processing have been used, to obtain more reliable information about obstacles in front of the ROV. Results from laboratory research work and from field experiments demonstrate that underwater obstacle avoidance with stereo cameras is possible and can increase the autonomous capabilities of ROVs by providing appropriate information for navigation, path planning, safer missions and environment awareness.

***Keywords*—underwater stereo vision, obstacle avoidance, ROV, AUV, motion tracking, position estimation**

I. Introduction

ROVs require a constant supervision during their missions and information about their environment. Even if underwater operations are characterized by low speeds, they can be very challenging due to the difficulty of not having a complete overview of the surroundings, due to the lack of light, due to non-linear distortions introduced by the water (relevant for visual sensors) and due to the impossibility to use high frequency radar because of high attenuation in salt water.
AUVs started to replace ROVs in several context, but due to limitations in the current artificial intelligence technology, their operative context is limited and ROVs are still broadly used today. ROV navigation is usually performed by one or more human operators that orient themselves with the help of a compass, a sonar and several cameras. By improving safety and success of the ROV missions a certain degree of autonomy within the ROV operations can be included. Cameras provide - compared with other sensors - cost effective information about the environment in high-resolution. That makes the implementation of such a system very interesting for obstacle avoidance, motion tracking, station keeping and drift correction.

A. Related Work

There are several sources in literature about obstacle avoidance and motion estimation utilizing vision for mobile and partial autonomous robots, but most of these studies are conducted for terrain, humanoid or aerial and not so much for underwater vehicles. An example of robotic vision and machine learning for autonomous underwater operations can be found in the field of visual servoing [1], [2].
Examples for underwater obstacle avoidance are in general based on multibeam echo-sounders [3] or other kind of sonars [4]. A previous work for underwater obstacle avoidance presents a structured light based visual subsystem, leading to the reconstruction of a 3D sea-line profile [5].
For motion estimation using cameras in the underwater environment the paper [6] can be considered. A collection of experiments is presented in [7] by testing a developed sensor fusion approach with a monocular camera and an Inertial Measurement Unit (IMU) [8]. Mosaicking techniques can also be exploited for motion estimation and are demonstrated in [9].

II. Hardware Setup

All field experiments where conducted using a ROV Sperre SUB-FIGHTER 30K (cf. Fig. 1). The ROV has been equipped with two Allied Vision Prosilica GC1380 cameras (CCD, global shutter, GigE Vision complaint, 1360x1024 pixels resolution), with a 8mm fixed focal length, an infinity focus and a maximum opening aperture. The cameras were placed inside two separate custom-made waterproof enclosures rated for 3000 meters depth. Two light sources (OSRAM 400W HMI) were utilized to illuminate the scene in front of the stereo system. Tests for obstacle avoidance and position estimation were performed at a depth of around 70 meters. The testing site we have chosen is the area around a shipwreck (“Hercules”, a fishing boat) in the Trondheimsfjorden. We have choosen a baseline of 9.5cm [10]. The cameras were synchronised using a software trigger and set with the standard factory

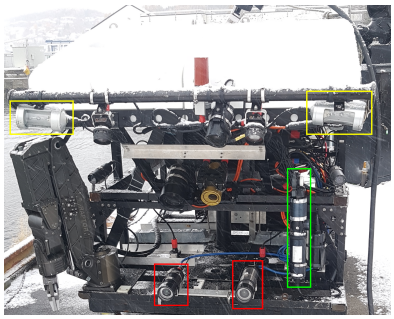

Fig. 1: ROV 30K: Stereo camera system (red rectangles), front looking sonar (green rectangle), light sources (yellow rectangles). See Fig. 2 for the final camera setup.

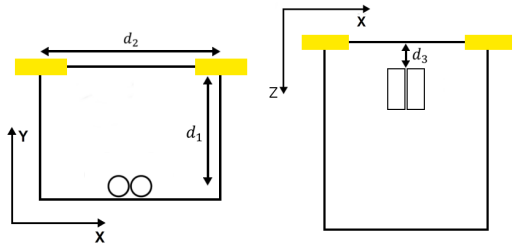

Fig. 2: Schematic drawing to illustrate the stereo camera - light source hardware setup at the ROV. **Left:** Front view, light sources (yellow), stereo camera (two circles), d_1=40cm and d_2=47cm. **Right:** Depth distance offset between the light sources and the cameras with d_3=20cm.

parameters except for the exposure time (15ms) and gain (15). In addition, we exploited a front looking sonar (Kongsberg Mesotech MS1000 with a high resolution head) for comparing the 3D information extracted from the stereo setup.
In the following we describe the tests we performed with respect to camera calibration, disparity map pre-processing, disparity map computation, 3D point cloud extraction and processing.

III. Camera Calibration

Camera calibration is a crucial step for extracting 3D information out of stereo images. Intrinsic and extrinsic camera parameters are computed along with a set of parameters that help to correct for lens distortions.
The intrinsic matrix contains the internal pinhole camera model parameters, and the extrinsic matrix describes the rotation and translation of the second camera with respect to the first camera, in a world coordinate frame. Typically, radial and tangential distortions are corrected. The radial distortion with parameters k_1, k_2, and k_3 can be described by the following equation

$$x_r = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_r = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$r^2 = x^2 + y^2.$$

Note that such distortions are commonly produced by standard lenses. The tangential distortion can be expressed as

$$x_t = x + [2p_1 xy + p_2((x^2 + y^2)^2 + 2x^2)]$$
$$y_t = y + [2p_2 xy + p_1((x^2 + y^2)^2 + 2y^2)]$$

with parameters p_1 and p_2, that appear in case the lens is not mounted parallel to the sensor. The elements within the intrinsic camera matrix

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

are the horizontal focal length f_x, the vertical focal length f_y, a skew parameter s (set if the camera pixels are not perfectly squared) and the components x_0 and y_0 representing the principal point offset. The rotation matrix R along with the translation vector t represent the extrinsic camera parameters. There are many methods available for determining these unknowns and we refer to [11] for an introduction to multiple view geometry.
We exploit image rectification in order to simplify the correspondence problem (the problem of finding matching points between stereo image pairs). When the intrinsic and extrinsic matrices are computed the homographies needed for rectifying a pair of images can be calculated algebraically. The first step is to calculate the fundamental matrix F, a 3×3 matrix of rank 2 that satisfies the following equation:

$$p'^T F p = 0.$$

Here p and p' are corresponding points in homogeneous coordinates between two cameras, so that Fp represents an epipolar line on which the point p' must lie in the other image [11]. The Fundamental matrix F can be calculated directly using the intrinsic matrices K and K' of the cameras along with the extrinsic rotation R and translation t

$$F = K'^{-T}[t]_\times R K^{-1},$$

where the operator $[\cdot]_\times$ denotes a matrix that performs the vector cross product. After F is obtained it's possible to determine the epipoles by finding the left and right null spaces of F

$$e'^T F = 0, \qquad Fe = 0.$$

This can be done by the Singular Value Decomposition (SVD) of the matrix F as follows

$$F = UDV^\top,$$

where U and V are two orthogonal matrices and D is a diagonal matrix. The i-th diagonal element of D is defined as the i-th singular value of F and the i-th column of U and the i-th column of V are the corresponding left singular vector and right singular vector, respectively of F. The right null space of F corresponds then to the column vectors of V that belong to zero singular values. The left null space of F corresponds to the rows of $U^\top$ with singular values equal to zero.

For the following rectification step it is required to find a projective transformation H' that maps the epipole e' to the point at infinity $(1, 0, 0)^T$. Then the matching projective transformation H, which maps the remaining epipole e to infinity is computed. Note that this also minimizes the least-squares distance:

$$\sum_i d(Hx_i, H'x'_i).$$

The rectification of the images itself is performed by re-sampling the two images according to the respective projective transformation H and H'.

Calibration of a stereo setup is a well studied subject and several toolboxes for different programming environments are available. The toolboxes used during our studies are the MATLAB integrated toolbox [12] (cf. Fig. 3), Caltech toolbox [13] and calibration functions from OpenCV [14].

When working with computer vision approaches for underwater applications the low contrast of images [15] impacts

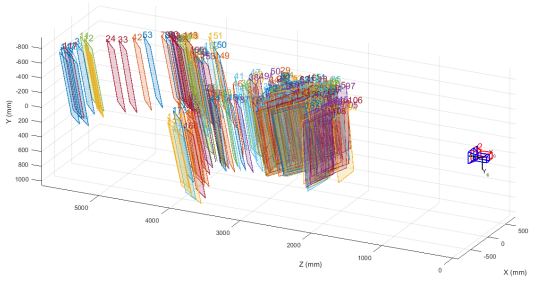

Fig. 3: Visualization of the reconstructed calibration patterns used in the stereo camera setup.

the ability to detect edges and corners, that is the first pre-processing step of many calibration toolboxes. Experiments showed that this problem is not significant and the automatic detection of the checkerboard worked well for images with reasonable quality (sharp enough and good illuminated, cf. Fig. 4). In order to assess the quality of the calibration results we

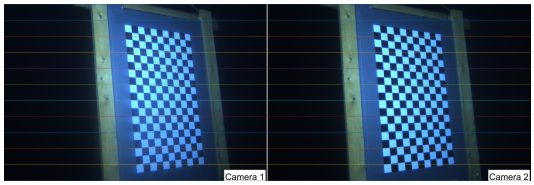

Fig. 4: Illustration of a rectification computed after calibration. Note that corresponding points are found on corresponding horizontal lines.

evaluated them in two steps, the first one was to compute the mean reprojection error. The reprojection error is defined as the distance in pixels between the actual projection of a calibration point in 3D and the reprojection of the reconstructed point onto the camera plane (cf. Fig. 5) using the 4×3 camera matrix $P = [R \mid t]^T K$. The reprojection error is mostly used

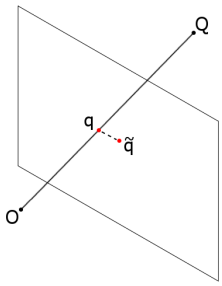

Fig. 5: Representation of the reprojection error: O camera origin, Q 3D point in space, q actual projection of the point Q on the camera plane, $\tilde{q}$ reprojected point. The reprojection distance is the euclidean distance in pixel between q and $\tilde{q}$.

to evaluate the result of a camera calibration, but at the same time it can be artificial lowered by iteratively removing images from the calibration images with the highest reprojection error (and it is also sensible to image resolution). As it may happen that the reprojection error is low, but the calibration is still sub-optimal (this occurs for example if the calibration pattern does not cover the full image [12]) we manually checked the rectified images for ensuring a good calibration.

The first tool that we tested for calibration was the integrated MATLAB toolbox. Two papers are cited in the toolbox documentation: The first one [16] describes a common method for camera calibration and is based on the identification of an arbitrary orientated planar pattern. The second paper [17] describes a calibration procedure, which consists of multiple steps: identification of a known pattern, a direct linear method for computing internal camera parameters that does not take into account lens distortion, followed by a Levenberg-Marquardt optimization in order to refine the results and calculate external parameters, taking into account the lens distortion correction. The calibration experiments were performed using two datasets: A dataset composed of 354 and a subset composed of 20 selected stereo-pair images. Using the integrated MATLAB toolbox we found a robust mean reprojection error of 1.08px and an average focal length of 1720.17px with an average estimation error of 0.09%. Removing all image pairs with a reprojection error of 2 pixels or more the average reprojection error reduces to 0.74px, the estimation error of the focal length reduces to 0.06% with an estimated focal length of 1719.36px. Note, that changing the optimization parameters did not impact theses results significantly.

The second calibration procedure that we tested was built from OpenCV calibration functions. OpenCV stereo calibration functions provide an automatic detection procedure of the checkerboard. By applying a histogram equalization to the gray value image and an adaptive threshold a binary image is generated. The corners of the checkerboard are detected with subpixel accuracy [18] using the respective OpenCV function (cornerSubPixel). A final optimization procedure is performed for estimating the lens distortions. In the used calibration example program the intrinsic camera matrices are estimated with

an initial guess. For optimization the Levenberg-Marquardt method is utilized with a termination criteria of 100 iterations or a delta change less than e^{-5}.
The calibration procedure using the Caltech calibration toolbox for a stereo setup starts with a manual independent calibration of the two cameras. For each camera the images have to be processed manually by providing the top left, top right, bottom left and bottom right corners of the calibration board. This process is not suited for processing a large amount of images, but allows to obtain a calibration when the automatic checkerboard detection fails due to non-optimal images (noise, non-homogenious illumination, etc.). The corners are then identified with subpixel accuracy using a Harris corner detector with a Gaussian mask.
The Caltech calibration toolbox estimates by default radial distortions (two coefficients) and the tangential distortions (two coefficients). The calibration process involves calibrating the two cameras separately before the stereo calibration procedure is called.
A satisfying rectification has been obtained with all three implementations. In order to evaluate the calibration results we compare the average reprojection error, the checkerboard recognition rate, the estimated focal length, the principal point, the rotation angle and the baseline for each calibration method.
In Table I and Table II f_x is the horizontal focal length, f_y

TABLE I: Calibration results full dataset

Parameters	OpenCV	MATLAB
$avg\ f_x$	1739.67px	1720.18px
$avg\ f_y$	1739.67px	1719.87px
$avg\ x_0$	676.39px	677.17px
$avg\ y_0$	506.54px	512.14px
x_d	-96.78mm	-100.13mm
y_d	-2.05mm	-2.56mm
z_d	-2.04mm	-12.16mm
α	$-1.32°$	$1.25°$
β	$-0.89°$	$2.41°$
γ	$2.69°$	$-2.29°$
Reproj. Err.	2.11px	1.08px
Check. recog.	58 pairs	196 pairs

is the vertical focal length, x_0 and y_0 are the coordinates of the principal point, x_d, y_d and z_d are the components of the translation vector, α, β and γ are the Euler angles around the x, y and z axis respectively.
The first experiment (cf. Table I) shows that the programm using OpenCV functions provides a lower detection rate of the checkerboard. The performance of the calibration in the OpenCV implementation that we used, in terms of the re-projection error, is poorer, but the translational displacements of the second camera relative to the first are more similar to those that we have physically measured. The biggest difference between the two implementations in proportional terms is the z_d parameter, MATLAB shows that the second camera is 12mm behind the first one, which is not true.
For the Caltech calibration we generated a subset of 20 images, since manual selection of the corners was too time consuming for the entire dataset. In this calibration run (cf. Table II) it is

TABLE II: Calibration results 20 stereo images

Parameters	OpenCV	MATLAB	Caltech
$avg\ f_x$	1683.73px	1715.78px	1721.29px
$avg\ f_y$	1683.73px	1718.26px	1721.54px
$avg\ x_0$	664.41px	699.70px	700.37px
$avg\ y_0$	511.13px	528.78px	520.00px
x_d	-102.16mm	-99.58mm	-101.34mm
y_d	2.07mm	-3.39mm	-1.77mm
z_d	1.67mm	6.81mm	-5.65mm
α	$2.49°$	$1.27°$	$-1.47°$
β	$-0.81°$	$2.31°$	$2.81°$
γ	$-1.34°$	$-3.06°$	$4.09°$
Re-proj. Err.	0.79px	0.75px	0.80px

possible to see that the MATLAB integrated toolbox and the Caltech toolbox tends to agree on many parameters, but again a strange behaviour in the offset z_d is present, and again all three implementations does not agree on the rotation angles, confirming that ideally a penalty term should be introduced during the parameter optimization forcing the angles close to zero. The tests that follow are performed using the MATLAB integrated calibration toolbox. We investigated the calibration results for different camera setups (cameras without enclosure, cameras inside the waterproof enclosures, and the stereo camera system underwater). The aim of these experiments was to determine which lens parameters are suitable for our specific application and to understand how these three different setups affect the calibration. Table III shows that the field of view

TABLE III: MATLAB Calibration results

Params	Underwater	Out. w. encl	Out. w.out encl.
Re-proj. Error	1.08px	0.40px	0.33px
Focal Len. px	1720.17	1286.23	1244.00
Focal Len. mm	11.13	8.32	8.05
H. FOV	$43.14°$	$55.74°$	$57.62°$
V. FOV	$33.03°$	$43.27°$	$44.83°$

decreases on average by 30% due to the refraction of water, but the combination of the frontal glass and water results in a decrease of the field of view of about 35%. This is a quite large change and has to be taken into account when working with cameras underwater.

IV. Disparity map calculation

In order to successfully perform visual obstacle avoidance underwater one has to estimate the 3D world coordinates of objects detected in front of the ROV, therefore, the next step after calibration and image rectification is to extract 3D information from the stereo recordings. This implies finding correspondences between the stereo image pairs. A disparity map represents this information and is a matrix of the size of the rectified image, containing in each element the offset between corresponding pixels (generally from the left to the right image).

Many algorithms have been proposed in the literature to solve the correspondence problem [19]. These methods can be classified into dense and sparse reconstruction, local, semi-global and global approaches. For the purpose of robot navigation, where limited computational resources are available and a fast reaction time with respect to the constantly changing environment is required, it's better to focus on computationally effective approaches. The algorithms that have been tested for the disparity calculation are the classical block matching (local approach), a semi-global block matching [20] and a global graph cut based algorithm [21], [22]. An open-source implementation of the graph cut based algorithm can be found in [23].

Block matching algorithms for disparity estimation calculate a score for a pixel involving it's neighborhood (generally the area involved in the calculation of this score is a square, so the terminology "block") and finding the block in the other image that has the most similar score. The matching is conducted exploiting the epipolar constraint, so that correspondences are searched along corresponding horizontal lines and the matching block is the one with the highest similarity measure. The similarity measure itself has an impact on computational efficiency and matching quality.

In our experiments we tested a simple and fast pixel block similarity measure known as Sum of Absolute Differences (SAD). Given a pair of square blocks of pixels A and B, both of dimension $n \times n$, the similarity measure is determined as follows:

$$SAD(A,B) = \sum_{i=1}^{n}\sum_{j=1}^{n} |a_{i,j} - b_{i,j}|.$$

We used the SAD block matching function from MATLAB with default parameter setting.

The semi-global block-matching approach [20] exploits an entropy based matching cost. The idea is based on a pixel-wise matching of Mutual Information and approximation of a global 2D smoothness constraint. The following energy function is minimized in order to obtain the disparity:

$$\begin{aligned} E(D) &= \sum_{p} C(p, D_p) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] \\ &\quad + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1]. \end{aligned}$$

The first term is the sum of all pixel matching costs for the disparities of D. The second term adds a constant penalty P_1 for all pixels q in the neighborhood N_p of p, for which the disparity change is maximal 1 pixel. The third term adds a larger constant penalty P_2, for all larger disparity changes [20]. Such a global minimization is a NP-complete problem and the solution is approximated by aggregating matching costs in 1D from all directions equally. The computational complexity of the algorithm is linear in the number of pixels and disparity range, obtaining an overall accuracy similar to global methods.

The aim of the Kolmogorov and Zabih stereo matching algorithm [21] is to minimize a non-convex objective function exploiting graph cut techniques. The energy for a match represented by f is given as follows:

$$E(f) = E_{data}(f) + E_{occlusion}(f) + E_{smooth}(f) + E_{unique}(f).$$

Here the data term measures how well a matched pair fits, the occlusion term minimizes the number of occluded pixels, the smoothness term penalizes the non-regularity of the configuration and the last term enforces the uniqueness of the match.

Due to the lack of ground truth data the disparity map evaluation has been performed by manually estimating a disparity map. Three scenarios were selected: A scene without an object in front of the ROV, a scene with a fish at the Hercules site and a scene with the Hercules site occluded partially by a dust cloud. We estimated the disparity map by starting with a 19x19 window sized semi-global approach. To ensure a good quality for the disparity map and to refine the contour of the objects, stereo anaglyph spectacles were used.

As our goal is obstacle avoidance we first needed only a rough distance estimate for an object in front of the underwater robot, as this already allows to plan and perform object avoidance actions. However, false object detections (noise) should be restrained in order to prevent unnecessary actions. We used the following quality measures for evaluating the computed disparity map: Root Means Square Error (RMSE), the percentage of "corrent" matchings (within a 3 pixel range) and the percentage of noise detected in the image. We define noise as the presence of a false match.

The first batch of experiments (cf. Table IV) were performed with block matching, first the uniqueness threshold is kept at 15 and the window size is changed, then the window size is kept fixed, where the best result (in terms of RMSE and obstacle detection rate) was reached and others parameters are varied. The second batch of experiments (cf. Table V) were performed using the semi-global approach, which turned out to be sensitive only to the window size. The graph cut (cf. Table VI) approach has been run, where the penalty parameter K (evaluating occluding pixels), $\lambda 1$ and $\lambda 2$ (smoothness parameters) and edge threshold were automatically determined. The presented results are computed as the average over the three test scene images. The rectified image size is 1168x788 pixels and the CPU used for this experiment was a Intel Core i7-5820K. The best performance has been achieved with the SAD block matching, with

TABLE IV: SAD Block matching results

W. size	Uniq. T.	RMSE	Obstacle Detec.	Noise	Time
7x7	15	27.84	22.11%	7.21%	**0.026**s
19x19	15	15.60	57.11%	0.25%	0.027s
55x55	15	14.36	**67.45**%	0.20%	0.162s
149x149	15	14.60	31.98%	0.25%	0.280s
55x55	-	55.47	25.50%	55.83%	0.153s
55x55	8	**12.94**	58.98%	0.81%	0.161s
55x55	21	14.90	65.38%	0.07%	0.158s
55x55	55	16.17	33.33%	**0**%	0.153s

TABLE V: Semi-Global Block matching results

W. size	RMSE	Obstacle Detec.	Noise	Time
7x7	28.50	33.42%	25.32%	0.45s
19x19	30.74	33.16%	27.61%	0.45s
55x55	15.65	28.41%	0.14%	0.45s
89x89	23.48	25.93%	4.22%	0.46s
149x149	18.87	21.83%	3.39%	0.49s

TABLE VI: Graph cut results

K	λ1	λ2	Data cost	RMSE	Obs. Detec.	Noise	Time
78.13	58.11	19.4	L2	76.75	0.01%	33.9%	432.5s

respect to computation time and obstacle detection rate. The semi-global approach provides a slightly more dense disparity map, but is less accurate and slower. The graph cut approach turned out to be too slow for real time obstacle avoidance. Note, that the execution time of the SAD block matching with a fixed window size depends only on the images size and the used disparity range.

The generation of the disparity map can be improved by pre-processing and post-processing operations like gap filling. Common pre-processing procedures for disparity map enhancement are homomorphic filtering and histogram matching. Homomorphic filtering can be applied to remove the slow changing illumination within an image, while preserving the high frequency component of the reflectance. Assuming that the reflectance and illumination is multiplicative within the intensity one remove the slow changing illumination by computing the logarithm of the image and applying high-pass filter to the result. Applying the inverse of the logarithm leads to an enhanced image. Histogram matching is a process were the histograms of two images (disparity map) are made similar, to homogenize the images so that a simple and fast matching procedure like SAD performs well. The algorithm calculates the cumulative probability distribution (cpd) of the reference image and the target image. The two cpd's are then used to build a look up table in order to update the intensity values of the target image. We repeated the disparity map tests after filtering the rectified images with homomorphic filtering, histogram matching and the two combinations of these. We observed that filtering did not improve the disparity map, given the fact that our performance indexes are not based on a ground truth, but on a manually refined disparity map, the performance fluctuations were not significant.

V. 3D INFORMATION EXTRACTION AND PROCESSING

By using the camera parameters we compute for each pixel with known disparity its corresponding 3D coordinates. The first step is to calculate the depth (z-coord.)

$$Z_p = \frac{fb}{x_{lp} - x_{rp}}, \tag{1}$$

where f is the focal length, b the baseline, and $x_{lp} - x_{rp}$ the disparity, i.e. the correspondence difference between the left pixel and the right pixel of point p. Then x and y coordinates can be computed as follows:

$$X_p = \frac{x_p Z_p}{f}, \quad Y_p = \frac{y_p Z_p}{f}. \tag{2}$$

False matchings (noise) will also appear as points in the 3D point cloud. This noise could be removed already in the disparity map, however noise filtering in the point cloud leads to better results.

One possible way to filter the point cloud is by applying a simple statistical filter [24]:

$$P_d = \{p_i \in P_{raw} | \, \|p_i - p_j\| > \mu + d_{thresh} \ \sigma \ \},$$

where P_d is the denoised point cloud, μ and σ are the mean and standard deviation of the nearest neighbour distances. Another possibility is to filter noise by point clustering. A subsequent noise filter can be applied to remove clusters with a low number of points, unstructured distribution, etc. We define a cluster-density based hypothesis by: The number of clusters and their shapes are unknown and we suppose that the cluster density is a function of depth, given the increased uncertainty in the stereo 3D reconstruction due to quantization errors [25]. However, in the literature various clustering algorithms are known, like centroid-based and/or distribution-based clustering relying on the knowledge of number and/or shape of clusters. Another more flexible density-based clustering approach, that also incorporates the presence of noise, is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) approach [26]. The parameters used by this algorithm are the point cluster density and the minimum number of points that can build a cluster. We note that the number of clusters is determined automatically. Hierarchical clustering is also suitable for this filtering problem, and we tested therefore an algorithm called Ordering Points To Identify the Clustering Structure (OPTICS) [27]. It is based on DBSCAN and addresses it's major weakness: the inability of finding meaningful clusters in data of varying density. OPTICS achieve the ability of finding meaningful clusters in data of varying density by putting the points that are spatially close to each other in a points list. Using this list a reachability-plot (a special kind of dendrogram), a hierarchical structure of clusters can be obtained. Input parameters for OPTICS are the minimum number of points per cluster and a search radius ε to consider point-to-point distances.

We analyzed our cluster-density based hypothesis by extracting clusters from all images we had (calibration and field test datasets). The disparity maps have been generated by block matching with the SAD metric using a window size of 7x7 (leading to noisy disparity maps) and an uniqueness threshold of 15. In order to reach real time performance the point cloud has been uniformly downsampled to 10% of the original amount of points. Clusters have been extracted with OPTICS with a minimum amount of 10 points and ε=0.05. The cluster density is the average distance of the cluster points from it's centroid. We are interested in removing noise within the 3D point cloud that could affect the quality of the path planning. We are looking for a noise removal process that removes points that do not belong to obstacles and, at the same time, keeps all relevant points belonging to obstacles. In order to evaluate the noise removal, we defined two performance indexes, the first one is related to the ability to not remove relevant obstacle points and is defined as the percentage of retained points within 0.1m from the points extracted from the estimated disparity map. The second performance index is related to the noise removal and is defined as the amount of removed points that are 0.1m further away from the points extracted from the estimated disparity map.

In addition to the noise removal that we discussed we added a modified version of DBSCAN to the test group that exploits prior knowledge (a empirical determined function about the clusters density, without compromising it's running time). The modification of DBSCAN consists in changing the input parameter ε to a function of depth of the currently evaluated

point. Note, the function $regionQuery(P, \varepsilon)$ returns all the

Algorithm 1 $DBSCAN\ for\ Stereo\ Point\ Cloud$

procedure $DBSCAN(D, MinPts)$
  C = 0
  for all *point P in dataset D* **do**
    if *P is visited* **then**
      continue
    mark P as visited
    NeighborPts = regionQuery$(P, f(P_z))$
    if *sizeof(NeighborPts)* $<$ *minPts* **then**
      mark P as noise
      continue
    else
      C = next cluster
      expCluster(P, NeighborPts, C, minPts)

Algorithm 2 $Subroutine:\ expandCluster$

procedure $expCluster(P, NeighborPts, C, MinPts)$
  add P to cluster C
  for all *point* P' *in NeighborPts* **do**
    if P' *is not visited* **then**
      mark P' *as visited*
      $NeighborPts'$ = *regionQuery*$(P',\ f(P'_z))$
      if *sizeof*$(NeighborPts')$ $>=$ *minPts* **then**
        NeighborPts += $NeighborPts'$
    if P' *is not yet member of any cluster* **then**
      add P' *to cluster C*

points with the euclidean distance smaller than ε from the point P including the point P itself. $f(P_z)$ is the empirical determined function that follows the found average cluster density in our experimental cluster density analysis.

Following we present an evaluation in terms of kept object points and correct noise removal of the discussed filtering approaches.

The point clouds are obtained from disparity maps generated with block matching and SAD metric. A set of 3 different window sizes (7x7, 19x19 and 55x55) and an uniqueness threshold of 15 was used to create the point clouds. The evaluation include DBSCAN ($\varepsilon = 0.15$, $minPts = 200$), OPTICS (same settings as DBSCAN), our modified DBSCAN ($minPts = 200$) and a statistical outlier removal approach [24] that evaluates 30 neighbors with $d_{thresh} = 1$ by applying a simple statistical filter. The results in Table VII show that

TABLE VII: Point Cloud Noise Filtering Results

Filter	Obj. Preservation	Noise Removal
DBSCAN	73.6%	89.5%
DBSCAN Stereo	96.3%	70.1%
OPTICS	53.7%	87.5%
Statistical	99.7%	45.2%

the performance of DBSCAN can be improved by using prior knowledge about how the cluster density changes (with a small cost of the noise removal). The statistical approach has shown great potential by preserving points belonging to the object, but it removes the least amount of noise. OPTICS does not perform well as it erodes the obstacles points. Hierarchical clustering assumes a hierarchical clusters structure (organization) that not always makes sense in this kind of clusters, and so the performance of OPTICS is the worst of the set regarding the preservation rate.

VI. Motion Tracking and Estimation

The stereo recordings were utilized to estimate the motion of the ROV by tracking features over consecutive image frames. Features from two consecutive images are first extracted, matched and then their 3D position is computed. An optimization algorithm provides the transformation that occurred between the image pair.

Features are extracted from the current stereo image pair (*currL*, *currR*) at time T_1 in addition to the features computed for the image pair (*prevL*, *prevR*) from the previous time $T_0 = T_1 - \Delta T$.

Features are detected in the images separately with the rotation and scale invariant Speeded Up Robust Features (SURF) descriptor [28] and then matched together spatially and temporally. Matching is performed using feature descriptors extracted from SURF and comparing their norm using the Sum of Squared Differences (SSD). We identify features that are present in all four images. These are then used to estimate the motion.

Let's denote with $\mathcal{F}_f$ the location of the features in frame f. The difference between matchings of features in frame $currL$ and frame $prevL$ is defined as

$$(du, dv) = \mathcal{F}_{currL} - \mathcal{F}_{prevL}. \tag{3}$$

Once we get such sets, we can use standard stereo reconstruction methods to estimate the three dimensional position of the features.

Given the sets of matched features found in the previous step, one can find the transformation (rotation and translation) which best describes the relation between the features at time T_0 and at time T_1. This can be formulated as an optimization problem (cf. also Figure 6) where one aims to find the minimum of an objective function defined as

$$f : \mathbb{R}^6 \to \mathbb{R} \qquad f(\underbrace{x, y, z, \theta, \alpha, \psi}_{\text{variables}}, \underbrace{P_{T_1}, \Pi, u_0, v_0}_{\text{given values}}) \mapsto \varepsilon, \tag{4}$$

where $\mathbf{x} = (x, y, z, \theta, \alpha, \psi)$ represents the motion (translation and rotation) of the camera between two frames at time T_0 and T_1, P_{T_1} are the 3D positions all the features detected at time T_1, $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection matrix used to map 3D coordinates onto the image plane. The variables (u_0, v_0) are the positions at time T_0 of all the features in the image plane. The median error of all N feature points is defined as

$$\varepsilon = \underset{i=1\ldots N}{\text{median}}\{(u_{0_i} - u_{e_i})^2 + (v_{0_i} - v_{e_i})^2\} \tag{5}$$

where (u_{e_i}, v_{e_i}) are the positions at time T_0 of the estimated features in the image plane as described in Figure 6. The optimal value is obtained as

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^6}{\arg\min}\, f(\mathbf{x}; P_{T_1}, \Pi, u_0, v_0). \tag{6}$$

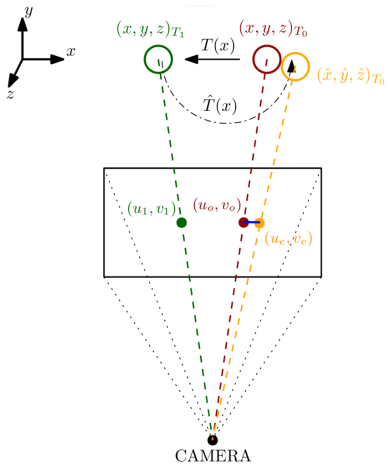

Fig. 6: Illustration of the minimisation problem (4). Example: Translation along the x-axis. Estimate the transformation $(T(x))$ between two frames, where only the 3D position of the features at time T_1 (green) and the position of the point in the image plane at both times T_0 and T_1 are known. We estimate the 3D position of the feature at time T_0 (orange) through the guessed transformation $\hat{T}$ and project it back to the image plane obtaining (u_{e_i}, v_{e_i}). The desired value of x is the one that minimises the distance between the two projections (blue segment).

To perform such a minimisation the *Nelder-Mead simplex method* has been used. It is a non-linear unconstrained model-free optimization procedure. The minimisation algorithm creates a polytope (simplex) on the variables space whose vertices sample the function at different locations, this simplex either expand or shrink in order to find the minimum.
The algorithm for estimating the camera motion between two frames is outlined in Algorithm 3.

We implemented two versions of the algorithm. The first one we call *(OME3)* [1] and it estimates only the translation and receives the differential angles as input while the other *(OME6)*[2] estimates all the six variables simultaneously ([29]) (cf. Fig. 9). The reason for that is, in some cases (especially with few features) the algorithm computes a transformation which is not the desired one but still explains the projected features quite well between the images. Note that equation (4) represents a non-convex optimization.

The camera pose (motion) estimation can be very challenging in some situations where the algorithm provides only a sub-optimal solution.

During lab experiments outside of the water we have seen that single (sparse) feature mismatching occurs regularly like shown in Figure 7.

[1]Optimised Motion Estimation - 3 parameters
[2]Optimised Motion Estimation - 6 parameters

Algorithm 3 Visual motion estimation

1: Get the stereo image pairs of 2 consecutive frames
2: Extract features
3: Match features in left and right images (spatial match)
4: Match them with the ones at previous step (temporal match)
5: Reconstruct these points which have been both spatially and temporally matched
6: Determine the camera transformation using Algorithm 4
7: Proceed to next frame and return to step 1

Algorithm 4 Pose estimation function

1: Use the previous time step's differential pose estimation as an initial guess
2: Build transformation matrix from previous two current frames
3: **for** $i = 1 \ldots N$ (N number of matched features) **do**
4: Solve $\hat{P}_{T_0} = \mathbf{T}P_{T_1}$
5: Project $\hat{P}_{T_0}$ to image plane to get (u_e, v_e)
6: Compute the error ε as defined in (5)
7: Use Nelder-Mead method to update angles and translations
8: **if** Error is sufficiently low **then** variables have been estimated correctly and thus the algorithm can stop
9: **else**
10: Proceed to next frame and return to step 2

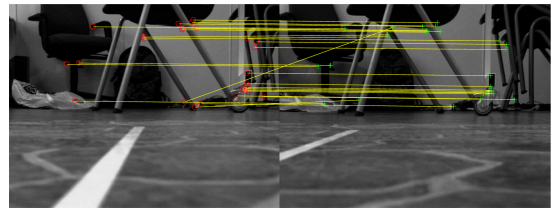

Fig. 7: Feature mismatching between left and right camera

Although such matching errors happen relatively due to the fact that features are tracked both temporally and spatially, they are still present in certain situations. Mainly when the environmental light is not good or texture regions look quite similar.

Such mismatchings can lead to an erroneous transformation estimation. However, since the objective function involves the median of the reprojection errors, outliers are not weighted. In Figure 8 features are drawn as red dots. The yellow lines indicate the larger matching positions in the previous frame.
For estimating the motion (between 2 frames) we exploited the rotation information provided by the IMU and estimated the differential translation with OME3. This result was refined by a following complete search finding all six variables (translation and rotation). From the obtained path in Figure 10 we can see that the estimated motion path agrees to a certain extend with the measured motion path. Over time the error accumulates and introduces a small drift. In all plots the blue line is the real trajectory while the red one is the estimated one.

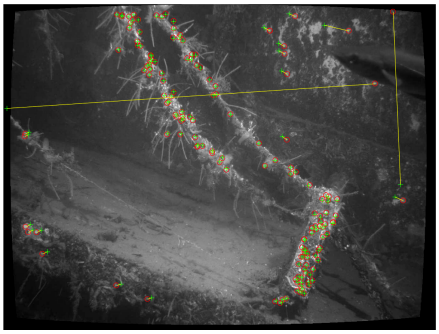

Fig. 8: Hercules site in Trondheimsfjorden: Feature matching result obtained in real time. Difference between real features (red) and the ones computed with the estimated transformation (green).

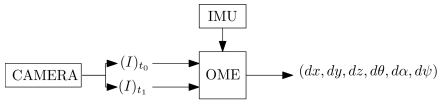

Fig. 9: Estimation algorithm running in mode (OME6)

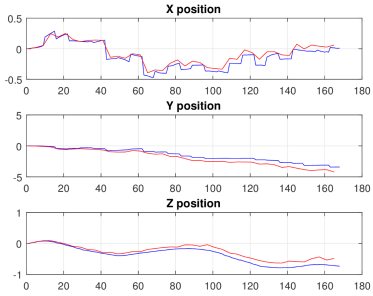

Fig. 10: Comparison of the estimated reconstructed trajectory (seconds, meters)

VII. Control System for Collision/Obstacle Avoidance

The stereo vision based collision avoidance system was implemented as a reactive part of an autonomy layer in the mission control system used for the autonomous ROV intervention [30]. The autonomy layer contains a deliberative module switching between several predefined behaviors in parallel with a system of reactive behaviors taking control of the vehicle in case of unwanted and unforeseen events, like an obstacle in the path of the vehicle. The system provides guidance input to the ROV control system. In our experiment, the vehicle was sent towards an obstacle known to the operator, in order to purposely test the detection of obstacles and to provoke reactive behavior for obstacle avoidance using robotic vision. When the obstacle in the point cloud (non empty), the orientation of the vehicle, and estimated distance to the obstacle is sent to the autonomy layer through User Datagram Protocol (UDP) communication. As a result, the reactive behaviour demands a change in heading of the vehicle. The vehicle turns until the obstacle is no longer detected in the stereo images, and a new way-point is defined straight ahead (cf. Fig. 11). During the

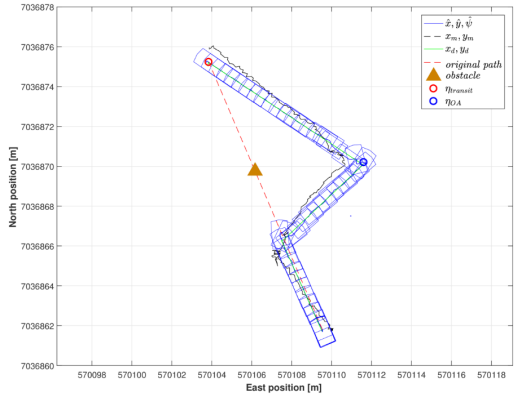

Fig. 11: ROV position in xy-plane during collision avoidance.

field testes it was confirmed that the 3D information coming from the stereo system were consistent with the information provided by the front looking sonar of the ROV. Note, fish were identified as obstacles. This problem can be addressed by a time analysis, clustering or machine learning approaches. Using MATLAB running on a i7-3820QM we were able to evaluate 5 point clouds per second, thus, given the slow speed of the underwater vehicle, we were able to achieve real-time performance. Real-time performance can be improved by reducing the resolution of the images in order to gain a higher frame rate.

VIII. Conclusion and Future Work

In this paper we presented a computer vision based ROV steering module that allows to add a certain degree of autonomy to a ROV mission. In particular we developed and analysed an obstacle avoidance module that is based on stereo-vision. It allows to detected obstacles in real-time and also determines the distance of the ROV to the object. Factors like turbidity and more difficult lighting conditions make the analysis of underwater video recordings in general more challenging than the analysis of video recordings in the air. Therefore we started with an analysis of three standard calibration implementations that are known to work well in air and exploited these for underwater stereo camera calibration. We found that the overall quality of the calibrations is lower comparing to calibrating stereo cameras in air but that all tested approaches in principle can be used to estimate the intrinsic and extrinsic parameters of the camera system. The task of computing a dense disparity map turned also out to be more challenging due to the lower quality of the calibration and due to the low sharpness of underwater images. However, fine tuning of a block matching approach with SAD has proven to be accurate

enough. Subsequent filtering of the extracted 3D point cloud of the scene has been achieved without any prior information by the means of statistical techniques and clustering methods, with the latter providing better results. Several tests with a ROV in the Trondheimsfjorden showed solid performance of the autonomous tracking and obstacle avoidance based on the stereo vision analysis even in absence of natural light. Future plans include the improvement of the underwater disparity map calculation and promising approaches are likely based on convolutional neural networks as they show for generic scenes a performance superior to other methods [31]. However, for a more accurate analysis and comparison of underwater disparity map calculations it is desirable to build an underwater ground truth data set for disparity maps.

ACKNOWLEDGMENT

This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU.

REFERENCES

[1] A. Carrera, N. Palomeras, D. Ribas, P. Kormushev, and M. Carreras, "An intervention-auv learns how to perform an underwater valve turning," in *OCEANS 2014-TAIPEI*. IEEE, 2014, pp. 1–7.

[2] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by girona500 uvms: A practical approach to autonomous underwater manipulation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 529–536.

[3] Y. Petillot, I. T. Ruiz, and D. M. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 240–251, 2001.

[4] E. O. Belcher, W. L. Fox, and W. H. Hanot, "Dual-frequency acoustic camera: a candidate for an obstacle avoidance, gap-filler, and identification sensor for untethered underwater vehicles," in *OCEANS'02 MTS/IEEE*, vol. 4. IEEE, 2002, pp. 2124–2128.

[5] G. Antonelli, S. Chiaverini, R. Finotello, and R. Schiavon, "Real-time path planning and obstacle avoidance for rais: an autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 216–227, 2001.

[6] J. Evans, P. Redmond, C. Plakas, K. Hamilton, and D. Lane, "Autonomous docking for intervention-auvs using sonar and video-based real-time 3d pose estimation," in *Oceans 2003. Proceedings*, vol. 4. IEEE, 2003, pp. 2201–2210.

[7] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with underwater robot localization and tracking," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 4556–4561.

[8] L. Armesto, J. Tornero, and M. Vincze, "Fast ego-motion estimation with multi-rate fusion of inertial and vision," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 577–589, 2007.

[9] R. Garcia, J. Batlle, X. Cufi, and J. Amat, "Positioning an underwater vehicle through image mosaicking," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2779–2784.

[10] J. Bercovitz, "Image-side perspective and stereoscopy," in *Photonics West'98 Electronic Imaging*. International Society for Optics and Photonics, 1998, pp. 288–298.

[11] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] "Matlab stereo calibration app," https://se.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html, accessed: 2017-01-11.

[13] "Matlab camera caltech calibration toolbox ," http://www.vision.caltech.edu/bouguetj/calib_doc/, accessed: 2017-01-11.

[14] "Opencv calibration toolbox," http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, accessed: 2017-01-11.

[15] M. Bryant, D. Wettergreen, S. Abdallah, A. Zelinsky *et al.*, "Robust camera calibration for an autonomous underwater vehicle," in *Proc. Australian Conf. on Robotics and Autom.* Citeseer, 2000, pp. 111–116.

[16] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[17] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition*. IEEE, 1997, pp. 1106–1112.

[18] R. Henkel, "Fast stereo vision with subpixel-precision," in *Proceedings of the sixth international conference on computer vision, Bombay*, 1998, pp. 1024–1028.

[19] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, 2015.

[20] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Computer Vision and Pattern Recognition. CVPR 2005.*, vol. 2. IEEE, 2005, pp. 807–814.

[21] V. Kolmogorov, P. Monasse, and P. Tan, "Kolmogorov and zabihs graph cuts stereo matching algorithm," *Image Processing On Line*, vol. 4, pp. 220–251, 2014.

[22] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 508–515.

[23] "Kolmogorov and zabihs graph cuts stereo matching algorithm implmentation," https://github.com/pmonasse/disparity-with-graph-cuts, accessed: 2017-03-14.

[24] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[25] R. Balasubramanian, S. Das, S. Udayabaskaran, and K. Swaminathan, "Quantization error in stereo imaging systems," *International journal of computer mathematics*, vol. 79, no. 6, pp. 671–691, 2002.

[26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[27] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[29] M. Dunbabin, K. Usher, and P. Corke, *Visual Motion Estimation for an Autonomous Underwater Reef Monitoring Robot*. Springer Berlin Heidelberg, 2006, pp. 31–42.

[30] T. O. Fossum, M. Ludvigsen, S. M. Nornes, I. Rist-Christensen, and L. Brusletto, "Autonomous robotic intervention using rov: An experimental approach," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–6.

[31] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1592–1599.

6.2 Paper B: Convolutional Autoencoder aided loop closure detection for monocular SLAM

Convolutional Autoencoder aided loop closure detection for monocular SLAM ⋆

Marco Leonardi * **Annette Stahl** *

* *Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway.*
e-mails: marco.leonardi@ntnu.no, annette.stahl@ntnu.no

Abstract: A correct loop closure detection is an important component of a robust SLAM (simultaneous localization and mapping) system. Loop closing refers to the process of correctly asserting that a mobile robot has returned to a previous visited location. Failing to detect a loop closure does in general not pose a threat to the positioning and mapping system of a robot, but a wrong loop closure can lead to drift of the robot and can therefore jeopardize the robot's mission. In this paper a robust, highly parallelizable standalone algorithm for globally detecting loop closures is proposed. The algorithm is purposely built with the goal of avoiding false positives, while maintaining reasonable true positives performances. Tests conducted on the KITTI and the Scott Reef 25 dataset show that when bag-of-words approaches perform poorly, our presented approach is able to avoid wrong loop closures.

Keywords: loop closure detection, monocular SLAM, loop closures, relocation system, error bounding, underwater navigation, underwater SLAM

1. INTRODUCTION

Loop closure detection is an important task linking together the localization and the mapping in the SLAM process. Loop closure detection makes it possible to reduce drift, perform relocalization when tracking is lost and allows to reconstruct the true topology of the scene.

Loop closure detection for visual SLAM is currently mainly based on bag-of-words models Engel et al. (2014) Mur-Artal et al. (2015) Cummins and Newman (2011), these have been proven to work very well in practice, but most of the time these algorithms are employed in indoor or urban environments. Early work in loop closure is linked to EKF-SLAM and the use of a *validation gate* [1] for data association Bailey and Durrant-Whyte (2006), which make use of the state estimation and can easily result in a wrong data association.

In this paper a method for loop closure that does rely only on images is presented, in this way a global identification of loop closures occurs, decoupling the drift in position estimation from the performance of the loop closure detection system. This method is independent from other components of a general SLAM system, allowing it to be further modified and plugged easily into a motion estimation and mapping algorithm to generate a SLAM algorithm. As this is an image-to-image method for loop closure detection that does not use any information about the estimated position at the time images are received, this method does suffer from the problem of *perceptual aliasing*, i.e. for equal sensed images captured at different positions within the environment (repetitive patterns in the environment), the algorithm will relocalize to the coordinates where the first image instance has been captured, thus indicating a wrong loop closure.

⋆ This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU

[1] After the prediction step in EKF-SLAM a prediction of the measurement is performed, this yields to a value-range in sensor space where to expect an observation. The area is called validation gate and is used to narrow the search and exclude other potential matches

2. RELATED WORK

Early attempts in the field of visual SLAM implemented loop closing, but not in a global way. The keyframe based SLAM method called PTAM Klein and Murray (2007) does provide loop closing based on the correlation of thumbnails of the keyframes, but this does not allow to perform large loop closures. Another popular keyframe based method for SLAM called ORB-SLAM Mur-Artal et al. (2015) is able to perform global loop closure detection, but the process is quite complex and involves many variables relative to the current state of the SLAM algorithm. An equally notable large scale monocular SLAM approach that provides global loop closure detection is LSD-SLAM Engel et al. (2014), which similarly to ORB-SLAM uses a bag-of-words approach for finding loop closure candidates.

One of the most notable works in appearance based localization is FAB-MAP as presented in Cummins and Newman (2008) Cummins and Newman (2011). The authors propose a probabilistic approach for solving the problem of recognizing places based on visual words and learn a generative model of place appearance.

Current state-of-the-art monocular SLAM approaches consist of depth prediction based methods exploiting con-

volutional neural networks Tateno et al. (2017) Ummenhofer et al. (2017). Such approaches, thanks to the learned priors about the shape of the objects, are able to provide a dense 3D reconstruction, with very low noise. Notably they are the first kind of monocular SLAM [2] that work in true scale [3], but still they don't provide any loop closure detection mechanics.

Williams et al. (2009) compares several types of loop closure detection for monocular SLAM: map-to-map, where correspondences are sought between features in two submaps, image-to-map where correspondences are sought between the latest frame from the camera and the features in the map and image-to-image, where correspondences are sought between the latest image from the camera and the previously seen images. They conclude that image-to-map systems perform best because such methods use as much information as possible, especially when combining image-to-image information and image-to-map information, but they have the problem of scalability. In a more recent survey Lowry et al. (2016) concluded that it is still a long way towards an universal place recognition system, but deep learning based techniques are the most promising at the current time.

3. APPROACH

The goal of this work is to build a standalone, global and highly reliable loop closure detection algorithm for monocular SLAM applications. The term standalone indicates the independence of the algorithm from other software and hardware systems in the robot. With the term global we indicate the ability to recover a loop closure independently from time and current position estimation. The term reliable means that the loop closure detection has to be robust and correct. For achieving a standalone algorithm only the images captured by the camera (and eventually the camera parameters) and the current single or multiple estimation of the position are exploited.
To achieve globality the information which can be obtained from the camera must, at least partially, retained within a database. Then the information or features coming from a new image has to be matched with the information stored previously in a database in order to detect a loop closure. To achieve reliability we employed a multi-step process where only images looking similar to each other were processed exploiting a robust point-wise matcher, which is robust regarding intensity changes.

A practical implementation of an algorithm would start by selecting the input images that will be further processed, since not every image contains useful information that can be exploited for re-localization (for example images where only a white wall is present). In addition, the image information database benefits from maintaining only relevant information, increasing its robustness and reducing the time needed for search operations. The processing then proceeds with an information extraction procedure followed by a matching procedure for identifying loop closure. After the matching attempt the obtained image information is also stored in the database, together with the single or multiple position estimation at the time when the image has been captured.

Considering related work, we can observe that feature based image matching methods have proven to be robust to scale, rotation, shift and illumination changes, but it's not computationally efficient to search for a similar image in a large database. Visual word based methods, thanks to the inverted index, provide a way to use features to search for similar images in a large database in a much more efficient way. However, one of the problems of visual words methods is that they require the generation of a vocabulary, which depends on the images supplied in the training phase. Another problem is that a match based only on features does not guarantee that the matched images belong to the same scene, given that matching features can be also present on partial and common repetitive patterns in the images. The main contribution of this paper is the use of a direct method that selects the images to undergo a further analysis with direct features matching, in order to avoid wrong image associations.

The direct method used is based on lossy image compression, which is achieved by reading the encoding layer of a convolutional autoencoder (CAE). CAEs are a kind of convolutional neural networks (CNN or ConvNet), which are deep, feed-forward artificial neural networks that have been used for feature extraction Masci et al. (2011) Geng et al. (2015) and image denoising Gondara (2016) Stowell and Turner (2015). The usage of a CAE still requires a training phase on a selected database, but as it will be shown, even a simple CAE is able to generalize very well, and so this method is able to provide loop closure candidates over very different types of images.

4. IMPLEMENTATION

In this section we describe the details of the implementation of the algorithm that was introduced above (sec. 3). In our implementation the images have been resized to 256×256 pixels. This choice was based on empirical tests with a convolutional neural network, that required images of fixed size as input. For higher resolutions the network did not converge properly within the training phase [4].

Here the proposed mechanism starts with the analysis of the output of a Canny edge detector applied to the resized image and the amount of non-zero pixels is calculated. The subsequent quality test consist in checking if the ratio between non-zero pixels and the total pixels of the image is larger than a threshold (4% in our implementation, the threshold has been chosen in accordance to the results of empirical tests, as it turned out that this threshold helps to avoid the further processing of images without enough information content). Images that have passed the quality test are then forwarded to two different processing stages: In the first SIFT (Lowe (2004)) is used to determine keypoints long with a descriptor, the second one is the convolutional autoencoder, which is used for generating a compressed representation of the images . The output of the most inner layer of the convolutional autoencoder

[2] Except inertial sensors aided visual SLAM
[3] The scale of the world cannot be observed and drifts over time in monocular SLAM, being one of the major error sources Engel et al. (2014)
[4] Machine learning problems are known to be sensible to the input space dimension Keogh and Mueen (2017)

(which in our implementation has a dimension of 4096) is then stored in an array together with the current best estimate of the position. Using a CUDA accelerated search with cosine distance metric the first 10 nearest compressed images are selected and their respective keypoint descriptor is retrieved from the database. Searching efficiently for similar data points in such high dimensions is an open problem in computer science Aggarwal (2001) [5]. In high dimensional spaces points essentially become uniformly distant from each other Aggarwal et al. (2001) when the Euclidean distance metric is used, and so different comparison metrics have to be employed. Generally these metrics do not have the strong and intuitive meaning that the Euclidean distance has. Exhaustive search operations are very inefficient on CPUs, as exhaustive search on CPU is an $O(n)$ operation, a CPU implementation is unlikely to work with real time performance. Given the simplicity and independence between each of the single operations of exhaustive searching, a parallel implementation on a modern GPU (that can run much more threads concurrently compared with a single CPU) keeps the observed search operation time close to constant. The cosine distance metric is one of the few metrics that can be applied in such a high dimensional space, and in this particular application it has proven to be a reliable metric. The keypoint descriptor of the current image is then compared to the keypoint descriptors of the 10 retrieved candidate images and finding 5 close descriptors indicate that the scene part has been seen before.

The strength of the algorithm is that the keypoint descriptor comparison is performed only between descriptors coming from images that are considered as similar by the autoencoder, thereby allowing an independent second verification of the scene similarity. In the current implementation the compressed image representation (obtained by the convolutional autoencoder) is finally inserted in the database.

In the following we describe the convolutional autoencoder (CAE) structure and training: The CAE that generates a compressed representation of the images, takes as input a RGB image with 256×256 pixels using normalized values in the interval $[0, 1]$. The encoding step starts with a convolutional layer based on 32 filters of size 3x3x3, and relu [6] as activation function, then max-pooling is executed with a pool size of 2x2. The same set of operations is repeated 3 times, each time with the output of the previous set of operations, with the only difference that the last convolution layer has only 16 filters, with the goal of further reducing the dimensionality of the encoding layer. To complete the autoencoder a decoding layer has to be implemented. A first thought is to perform an inverse operation of the pooling. Pooling is a sampling process that involves loss of information, so inverting it can involve zero-filling or interpolation, preventing such information from being completely recovered. While there is a strong consensus in the deep learning community about the superior performance of the max-pooling Scherer et al. (2010), there seems to be not such consensus about how to perform unpooling, so the strategy employed in this work is just a simple resizing with nearest neighbor interpolation. After the resizing layer a convolutional layer based on 32 filters of size 3x3x3 is in line, and relu as activation function, then again another equal resizing layer. The final structure of the network consists of a convolutional layer as previously presented and another convolutional layer where the numbers of filters is equal to the original image dimensionality. In this convolutional layer the activation function is a sigmoid function, that also represents the output of the network for the training phase. A zero-padding strategy is employed for every operation that involves sliding window operations, like convolution and pooling. At the start of the training procedure all the weights are initialized by sampling from a zero mean Gaussian distribution with standard deviation of 0.05. The training procedure involves minimizing the error between the target and the actual output of the network through a sigmoid cross entropy given logits [7] function, which allows the network to perform multi-class classification (see Google Tensorflow documentation (2018)), to act as an autoencoder. The optimization is done through a first-order gradient descent method, based on adaptive estimates of lower-order moments called ADAM Kingma and Ba (2014), which has been found to be one of the most successful optimization strategies by the deep learning community Goodfellow et al. (2016). Even if ADAM is a gradient descent method that is able to adjust the learning rate, it still needs a reference parameter for it, which has been set to 0.001. The dataset consists of 300 images of indoor and outdoor environments captured from a handheld camera. The dataset was split in 70% training and 30% validation examples/images. The optimization was performed on mini-batches of 32 images at the time for 5000 epochs. At every epoch the loss on the validation set is calculated, finally the network weights which performs best on the validation set represent the final outcome of the training procedure.

5. EXPERIMENTS

Performance analysis in SLAM is a complex issue, especially when it comes to a stability analysis of the SLAM algorithm. Therefore, they are often produced by simulations or extended field tests. In order to benchmark the SLAM algorithm a selection of the KITTI dataset Geiger et al. (2013) is used, which contains large sequences of images coming from front looking stereo cameras mounted on a car. We tested our loop closure detection algorithm on the KITTI dataset (sequences 00 and 02) using only the left images. As we plan to use this loop closure detection method for underwater SLAM, we tested it also on an underwater SLAM dataset Scott Reef 25 from the Australian Centre for Field Robotics mar (2009). Given that the dataset is composed of stereo images, we choose again to use the left images.

For comparison we tested the FAB-MAP Cummins and Newman (2008) on the same datasets. There exists an improved version of it called FAB-MAP 2.0 Cummins and Newman (2011), but the implementation is not freely

[5] All current indexing techniques (based on space partitioning) degrade to linear search for sufficiently high dimensions Datar et al. (2004) Weber et al. (1998) Gionis et al. (1999)

[6] Rectified linear unit: $f(x) = x^+ = max(0, x)$

[7] Logits are functions that maps probabilities $x \in [0, 1]$ to $\mathbb{R}$ $y \in (-\infty, \infty)$

available to the public. FAB-MAP has been run with default parameters, both our algorithm and FAB-MAP have been run by accepting as loop closure candidates only frames that have at least 9 frames inbetween. As FAB-MAP provides a probabilistic value for each couple of frames to represent a loop closure, it's needed to choose a threshold for asserting which of the images represent a loop closure, this threshold has been set to 99%. We used for the tests an Intel Core i7-5820K with 32GB of RAM and a Nvidia TITAN GPU with 6GB of GDDR5.

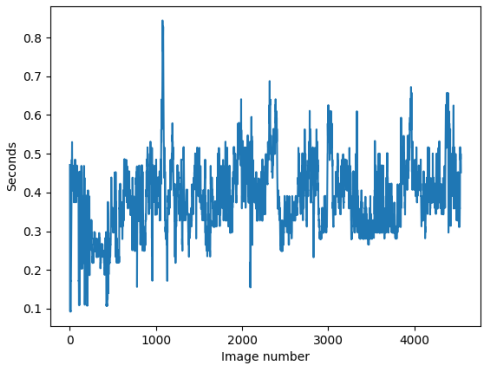

Fig. 1. Required time in seconds for processing a new image plotted against the images of KITTI dataset sequence 00.

Table 1 shows the results of the experiments. We observe a superior performance in terms of correctness of the identified loop closures on all the datasets, while FAB-MAP performs better in terms of computation time. The low performance of FAB-MAP on the underwater dataset is probably due to the fact that the used vocabulary is not adequate for the environment. Tests with different values for the main parameter of FAB-MAP (the true positive rate $p(z = 1|e = 1)$, that represent the probability of observing a feature given the existence of that feature) show little to no effect on the performance. Figure 1 shows a plot of the computation time for the images present in the sequence 00 of the KITTI dataset. The computation time of our solution can be approximated as a constant. Currently the algorithm may detect wrong loop closures candidates, like shown in figure 2 based on the observation and detection of single similar images in the video stream/sequence. This *perceptual aliasing* can be reduced or avoided if we also consider temporary information in the algorithm performing an additional consistency check for identified loop closure candidates.

Regarding the memory consumption of the algorithm, it is notable that each stored image occupies 16384 bytes (4094 32bit floats), assuming a frame rate of 30 frames per seconds and that every image pass the test of information content, the algorithm will allocate around 1.65 GB/hour of GPU memory. Newer GPUs support 16 bit float natively Ho and Wong (2017), which will immediately halve the memory consumption without any measurable performance impact on the algorithm.

Dataset	**Images size**	**No. of images**	**Autoencoder** described params prec., time, loops	**FAB-MAP** $P = \mathbf{0.35}$ prec., time, loops	**FAB-MAP** $P = \mathbf{0.39}$ prec., time, loops	**FAB-MAP** $P = \mathbf{0.45}$ prec., time, loops
KITTI, sequence 00	1241x376	4541	99.7%, 1782s, 848	-	99.4%, 1226s, 344	-
KITTI, sequence 02	1241x376	4661	100%, 2189s, 302	-	94.5%, 1222s, 73	-
Scott Reef 25	1360x1024	9831	100%, 8825s, 15	21.1%, 7446s, 119	23.4%, 7403s, 111	22.7%, 7873s, 119

Table 1. The table shows the results of the loop closure detection experiments. P is $p(z = 1|e = 1)$ in FAB-MAP, that corresponds to the P_OBSERVE_GIVEN_EXISTS parameter in the FAB-MAP configuration file.

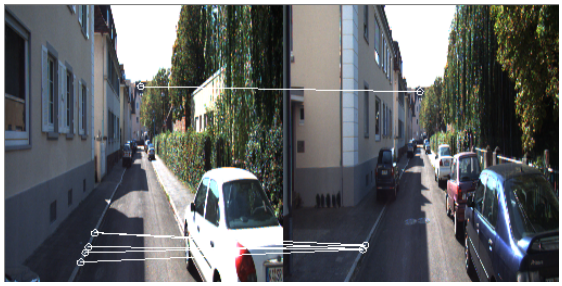

Fig. 2. Sub-optimal loop closure detection due to *perceptual aliasing*.

6. IMPROVEMENTS AND TAILORING

In this first implementation the memory on GPU is allocated and deallocated everytime a set of similar images has to be retrieved. This does take most of the computation time required for this operation. A more efficient solution would be to just add to the GPU memory the new images compressed by the autoencoder.

The convolutional autoencoder has great potential for improvements and further research must be conducted for finding an optimal network architecture for encoding images. Especially from a training point of view, for instance a multi step training would be beneficial, because even ADAM does still suffer from the vanishing gradient problem. Further parallelism can be achieved by splitting the image in multiple patches in order to run in parallel multiple instances of a single autoencoder that, due to the lower input dimensionality, would be easier to train.

Feeding inertial measurements to a Kalman Filter can provide a state covariance with a guaranteed physical meaning of state uncertainty (as the measurement error is bounded within a certain time period), so restricting the search of potential matches using the state covariance can improve robustness and speed up the search process.

As previously stated in the current implementation, at the end of the algorithm the compressed representation of the current image is added to the database. A more efficient solution would be to store a measure of the position state/keyframe uncertainty (a measure could be the magnitude of the state covariance matrix if an EKF is used for position estimation). And, after a loop closure, eventually replace the matched image in the dataset if the state uncertainty that comes with the new image is lower than the matched image currently in the dataset.
It has to be noted that together with position estimation also attitude could be stored, but current research in nonlinear observer theory has produced globally exponential stable estimators for attitude estimation Grip et al. (2015) and as IMUs today are very cheap, very small, and present in almost every electronic device, attitude estimation using computer vision is, for most of the applications, no longer necessary.

7. CONCLUSION

In this paper a global, standalone loop closure detector for monocular SLAM has been presented. Tests conducted on relevant datasets known to the SLAM community show very good performances in correct loop closure detection, with comparable computation time given parallelization on GPU.

Using compressed image representations for selecting which images attempt to match with direct features does indeed guard the loop closure detection algorithm from matching images given features that lie on repetitive patterns, and also provide a way to match directly feature descriptors instead of using visual words.

It has to be noted that our approach does not allow to detect loop closures with a large variance of the view points, but is exact where other loop closure detection systems are more likely to fail.

This work follows the direction of analyzing the use of deep learning for SLAM purposes, given that the SLAM community has found deep learning techniques to be currently irreplaceable for complex SLAM operations.

ACKNOWLEDGEMENTS

We would like to thank the Autonomous Vision Group part of the University of Tübingen for providing the KITTI dataset and the Australian Centre for Field Robotics marine robotics group for providing the Scott Reef 25 dataset. We would like to thank Trym Vegard Haavardsholm for his help in reviewing this work.

REFERENCES

(2009). Marine robotics datasets, australian centre for field robotics. http://marine.acfr.usyd.edu.au/datasets/. Accessed: 2018-01-15.

Aggarwal, C.C. (2001). On the effects of dimensionality reduction on high dimensional similarity search. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 256–266. ACM.

Aggarwal, C.C., Hinneburg, A., and Keim, D.A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, 420–434. Springer.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3), 108–117.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6), 647–665.

Cummins, M. and Newman, P. (2011). Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9), 1100–1123.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V.S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, 834–849. Springer.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

Geng, J., Fan, J., Wang, H., Ma, X., Li, B., and Chen, F. (2015). High-resolution sar image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 12(11), 2351–2355.

Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *Vldb*, volume 99, 518–529.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*, 241–246.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Google Tensorflow documentation, T.d. (2018). Sigmoid cross entropy given logits. https://goo.gl/xeXVSY. Accessed: 2018-02-19.

Grip, H.F., Fossen, T.I., Johansen, T.A., and Saberi, A. (2015). Globally exponentially stable attitude and gyro bias estimation with application to gnss/ins integration. *Automatica*, 51, 158–166.

Ho, N.M. and Wong, W.F. (2017). Exploiting half precision arithmetic in nvidia gpus. In *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, 1–7.

Keogh, E. and Mueen, A. (2017). Curse of dimensionality. In *Encyclopedia of Machine Learning and Data Mining*, 314–315. Springer.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 225–234.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2), 91–110.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., and Milford, M.J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1), 1–19.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, 52–59. Springer.

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

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, 92–101. Springer.

Stowell, D. and Turner, R.E. (2015). Denoising without access to clean data using a partitioned autoencoder. *CoRR*, abs/1509.05982.

Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). CNN-SLAM: real-time dense monocular SLAM with learned depth prediction. *CoRR*, abs/1704.03489.

Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2017). Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5.

Weber, R., Schek, H.J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, 194–205.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12), 1188–1197.

6.3 Paper C: Deep learning based keypoint rejection system for underwater visual ego-motion estimation

Deep learning based keypoint rejection system for underwater visual ego-motion estimation ⋆

Marco Leonardi * **Luca Fiori** ** **Annette Stahl** *

* *Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway. e-mails: marco.leonardi@ntnu.no, annette.stahl@ntnu.no*
** *Department of Information Engineering and Mathematics, University of Siena, Italy, e-mail: luca.fiori@student.unisi.it*

Abstract: Most visual odometry (VO) and visual simultaneous localization and mapping (VSLAM) systems rely heavily on robust keypoint detection and matching. With regards to images taken in the underwater environment, phenomena like shallow water caustics and/or dynamic objects like fishes can lead to the detection and matching of unreliable (unsuitable) keypoints within the visual motion estimation pipeline. We propose a plug-and-play keypoint rejection system that rejects keypoints unsuitable for tracking in order to obtain a robust visual ego-motion estimation. A convolutional neural network is trained in a supervised manner, with image patches having a detected keypoint in its center as input and the probability of such a keypoint suitable for tracking and mapping as output. We provide experimental evidence that the system prevents to track unsuitable keypoints in a state-of-the-art VSLAM system. In addition we evaluated several strategies aimed at increasing the inference speed of the network for real-time operations.

Keywords: keypoints, monocular SLAM, VSLAM, Visual Odometry, VO, underwater navigation, underwater SLAM, noise filtering

1. INTRODUCTION

Visual odometry and VSLAM have been successfully applied in the underwater domain, as for example for ship-hull inspection Kim and Eustice (2013), Kim and Eustice (2009) or autonomous underwater navigation Bonin-Font et al. (2015). Other previous work relies mainly on inertial navigation system (INS) measurements Krys and Najjaran (2007), while a very modern work proposes feature based visual odometry Ferrera et al. (2019). The interest in this topics is continuously growing and lately a dataset dedicated to real-time visual underwater localization has been published by Ferrera et al. (2018). In addition, many SLAM systems for dynamic non-underwater environments have been presented by Bescos et al. (2018), Cui and Wen (2019), and Tan et al. (2013). Unfortunately most of these systems are very sensitive to moving objects in the scene. Therefore, our system aims to prevent keypoints detected at moving objects or dynamic textures, which in a state-of-the-art approach might have been considered for ego-motion estimation and mapping. Keypoints are prevented by using the prior learned by a neural network, so that, depending on the VO/VSLAM system, the tracking is not affected by the presence of moving objects or dynamic textures in the scene.

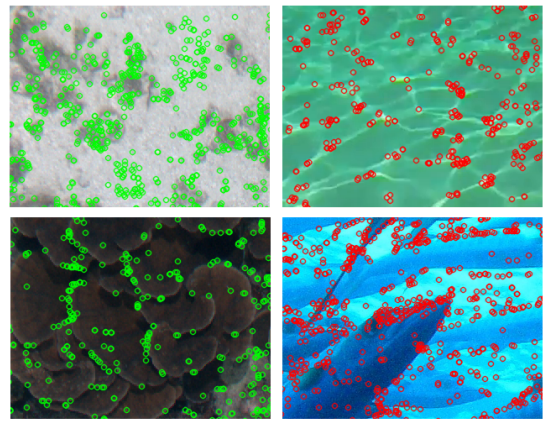

Fig. 1. Keypoints labeled with our system: Keypoints classified as *suitable* are circled in green, keypoints classified as *unsuitable* are circled in red. Top left: sandy seabed, top right: caustics, bottom left: seabed with vegetation, bottom right: a *school* of fishes.

We provide experimental evidence of this statement in the result section (section 4).

⋆ This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU.

Fig. 2. Example of *unsuitable* (left) and *suitable* (right) keypoint image patches. *Unsuitable* image patches can contain fishes, a crab, seaweed, caustics, clean water with high gradient zones, and the effect called *marine snow* (first row, third image to the right). *Suitable* image patches can contain for example man-made objects and different sea bed types.

We extract patches centered around each detected keypoint as input for a convolutional neural network (CNN) and train it to distinguish if the keypoint is *suitable* for tracking or not. We define keypoints as *suitable* which are detected at (static) surfaces, that are not supposed to move (e.g. rocks), and exclude those keypoints - defined as unsuitable - which are detected at moving objects, textures or dynamic physical phenomena (eg. fish, caustics, etc.; see Fig. 1). Thereby, keypoint neighborhood image scene information serves as context for the CNN, which is trained to distinguish if the detected keypoint belongs to a moving surface or not. This prior knowledge based selection makes it possible to forward only keypoints to a motion estimation pipeline that are exploitable for a reliable ego-motion estimation.

The two main contributions presented in this paper are:

- A fast supervised way to generate a dataset for keypoint classification
- A novel method for reliable keypoint selection for underwater visual ego-motion estimation

1.1 Related Work

In Wangsiripitak and Murray (2009) a parallel implementation of monoSLAM was presented, that incorporates a 3D object tracker into the SLAM system. Moving features are thereby not included into the map and features that are known to be occluded by objects are deleted, but as this work does not detect moving objects, non-stationary features apart from the ones laying on the tracked moving object will still corrupt the SLAM estimation. In a similar way, Riazuelo et al. (2017) implemented an object tracker dedicated to people tracking. These methods would detect those *a priori* dynamic objects, but fail to react on movements of *a priori* static objects.
In Tan et al. (2013) the authors compare features between keyframes and the current frame for 3D structure validation. This method fails when *a priori* dynamic objects remain static (like e. g. lifeless marine fauna).
Recently, an ORB-SLAM Mur-Artal et al. (2015) based approach was presented that operates reasonable well in dynamic environments Cui and Wen (2019). While ORB-SLAM is already implicitly robust to small changes in the scene, the authors improve the robustness of the algorithm by comparing image patches around the re-projected 3D point between the current frame and the reference frame. However, while the system is reported to increase the tracking performance in dynamic environments, it's not able to cope with scenes where many dynamic objects with a coherent motion pattern are present (e.g. a school of fishes). The authors suggest the use of object recognition techniques to improve the robustness, since dynamic scenes can generate a valid apparent ego-motion.
Similar to the work presented in this paper DynaSLAM Bescos et al. (2018), and ORB-SLAM2 Mur-Artal and Tardós (2017), a monocular approach uses Mask R-CNN He et al. (2017) which is state-of-the-art for object instance segmentation. The deep network architecture is designed to classify and to reject keypoints associated with regards to a list of dynamic object labels (i.e. person, bicycle, car, etc.). Unfortunately, DynaSLAM doesn't work in real-time, as Mask R-CNN alone runs at around 195 ms per image on a Nvidia Tesla M40 GPU He et al. (2017). Only a few underwater labeled datasets are available, while to the best of our knowledge no datasets exist with respect to semantic segmentation OByrne et al. (2018).
Very recently, an underwater VO and feature-based SLAM approach Ferrera et al. (2019) was published not taking into account the dynamics of underwater environments, exposing the system to be highly sensible to fishes, shallow water caustics and other dynamic objects present, including seaweed transported by current.
Our method falls into the category where keypoints laying on *a priori* dynamic objects will always be removed, even if the objects are most likely going to remain static for the time they are going to be observed (e.g. lifeless marine fauna). We argue that the *a priori* dynamic objects are extremely unlikely to remain static in the underwater environment, and that these objects should never be used

for Visual SLAM as they could be only momentarily static (e.g. a fish resting on the seabed).

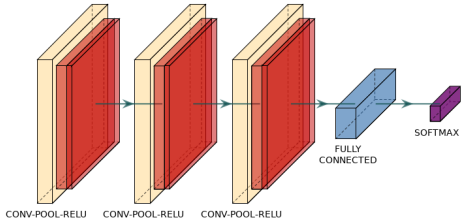

Fig. 3. Representation of the network architecture: the first three layers are a stack of CONV-POOL-RELU layers, extracting features for the fully connected layer. The output layer is a softmax layer.

Compared to methods that use deep neural networks for object-instance segmentation, our approach does not need to create masks for the training procedure, and the overall execution time is also positively affected, thanks also to the highly-parallelizable nature of the problem. Patch creation and analysis is not directly tied to image resolution, making a network trained in this way more future-proof.

2. DATABASE CREATION

We composed our dataset of underwater images from the following datasets: the *Tasmania Coral Point Count*, the *Scott Reef 25* and the *Tasmania O'Hara 7* from the Australian Centre for Field Robotics for Field Robotics (2009), the *An Underwater Observation Dataset of Fish* Eickholt (2018) and images from the *Herkules wreck site* coming from several field surveys the authors conducted Leonardi et al. (2017). For our tests 110 images out of this dataset were selected and in total 13158 patches were extracted (see Fig. 1), where 60% were used for the training set, 20% for the validation and 20% for the test set. Possible underwater dynamic image scenes are presented, including various fish species, crabs, algae, floating particles, as well as illumination changes and effects and man-made floating objects.
The dataset is created utilizing a manual labelling procedure. For each image, up to 1000 keypoints are extracted using the Oriented FAST and Rotated BRIEF (ORB) Rublee et al. (2011) feature detector by setting the FAST threshold to 20, which is a commonly used value.
Images are proportionally scaled, keeping the aspect-ratio, considering the first image as reference; this process aims to consistently scale context information in the patches centered in the keypoints.
We implemented a software interface to ease the process of differentiation between *suitable* keypoints (laying on *a priori* static objects) and *unsuitable* keypoints (laying on *a priori* dynamic objects), see Fig. 4. Our software visualizes the image and shows the ORB extracted keypoints as dots. In the interface the user can draw multiple independent polygons as successions of segments. Keypoints inside the polygons are marked *unsuitable*, while keypoints outside the polygons are marked as *suitable*. Image patches of 257×257 pixels (cf. Fig. 2) are extracted around each keypoint. The size of the patches has been selected in a process of trial and error, with the goal of jointly maximizing the contextual information and the inference time performance of the resulting network. All patches corresponding to *suitable* keypoints are stored separately from those corresponding to *unsuitable* keypoints.

3. NETWORK ARCHITECTURE

In this section we describe the details of the network architecture as illustrated in Fig. 3. In our implementation the keypoint patches have been resized to 65×65 pixels. This choice was based on empirical tests, since the patches where originally 257×257 pixels to preserve context information. For a higher resolution (129×129 and 97×97 pixels) of the patch the network did not gain in precision, instead decreasing its performance in both memory and speed. We recommend, that the size of the extracted rectangular patches should be around 2/10 of the image width, considering a maximum aspect ratio of 16:9. The idea is that it should be possible for a human operator to visually analyze and discriminate successfully the patches. The proposed network is a convolutional neural network, counting three *convolutional* (CONV) layers, one *fully-connected* (FC) layer and a *soft-max* layer.
The first layer is a CONV layer, consisting of 16 filters with a kernel size of 3×3, stride of 1 in each direction, zero filling padding strategy, followed by a 2×2 *max-pooling* (POOL) and a *rectified linear unit* (RELU) layer for adding non-linearity. The second and third layer are identical to the first one. It follows a *flatten* layer which is needed to format the input for the following FC RELU layer. The FC layer (128x128) is followed by a soft-max RELU layer with two outputs, representing the *suitable*/*unsuitable* posterior class probabilities, given the input and the network configuration. The neural network architecture is inspired by the original LeNet-5 LeCun et al. (1998) network, with a different amount of layers and a different layer dimensioning due to the more complex discrimination problem. Different network topologies and configurations have been tried, the one we just presented is the network that provided us with the best ratio between the amount of network weights and classification performance.

3.1 Training procedure

The training of the network was performed over the dataset described in section 2, with the patches resized to 65×65 pixels. The *batch size* was set to 64 and the network is trained for 1000 epochs. The set of weights which is retained is the one that performs best in terms of validation loss. The optimizer used is adaptive moment estimation (ADAM) Kingma and Ba (2014), with a learning rate of 0.001 and $\epsilon = 1e-10$. The loss function used in both training and validation is the mean square error (MSE). The target follows the *one hot encoding*. In this way the inference output of the network approximates posterior probabilities Richard and Lippmann (1991). We obtained an accuracy of 96.7% on the test set. In Table 1 the final confusion matrix is presented.

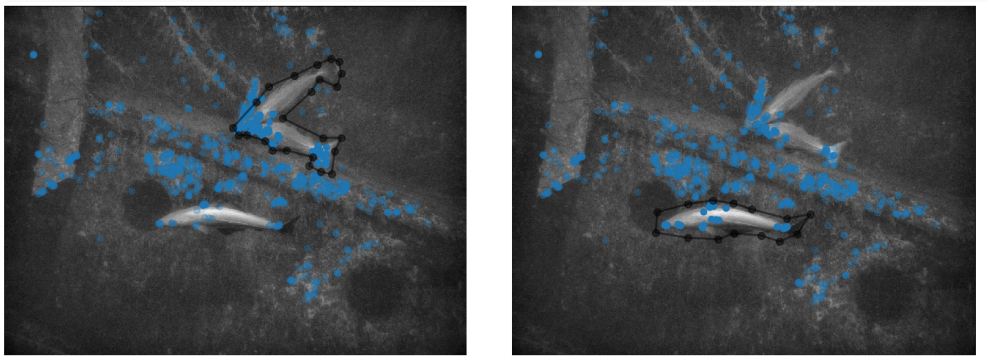

Fig. 4. Software interface procedure for the selection of *unsuitable* keypoints. During this procedure, not all of the *unsuitable* keypoints are selected by drawing only one polygon. As soon as the first polygon is selected, the procedure asks to eventually draw another polygon. In the picture on the left, first a polygon is drawn to enclose the keypoints on the fishes in the top-right quadrant, then in the picture on the right a second polygon is drawn to select the keypoints laying on the remaining fish.

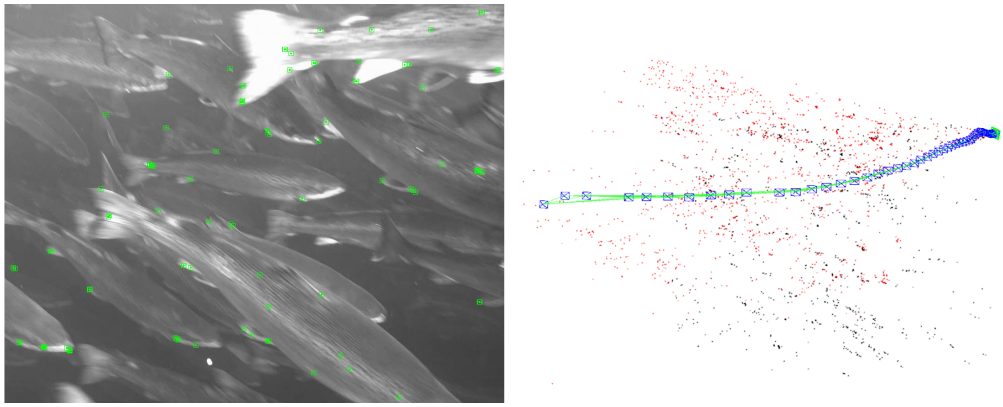

Fig. 5. Example of ego-motion estimation through a fish swarm present in the LAKSIT sequence 1, generated by monocular ORB-SLAM with a tuned initialization procedure as described in this paper. On the left the current image with the tracked keypoints and on the right the 3D map with the camera poses (in blue) with the covisibility links (in green). Initialization, mapping and tracking operations that follow rely entirely on keypoint correspondences that are laying on moving objects. To be noted that keypoints highlighted in green on the left part of the image are the keypoints currently tracked by ORB-SLAM, here they are not indicating *suitable* keypoints found by our neural network.

Patches: 2631	Pred. Suitable	Pred. Unsuitable
Suitable	1532 - TP	19 - FN
Unsuitable	68 - FP	1012 - TN

Table 1. Confusion matrix calculated over the test set (cf. section 2): True positive (TP), True negative (TN), False positive (FP), False negative (FN). Percentages: FP 2.6%, FN 0.7%

4. RESULTS

In this section we present results in terms of performance in drift reduction and in terms of inference speed, with different execution configurations and different network floating point precision. During all the tests no pre-processing, like histogram manipulations or brightness augmentation, is performed. For analyzing the capabilities of drift reduction, we analyzed several footages recorded by a static camera inside a fish cage, used at a fish farming site (LAKSIT project Føre, M., Frank, K., Svendsen, E., Schellewald, C., Sunde, L.M., Alfredsen, J.A. and Stahl, A. (2017), to this date the dataset is not publicly available, but it may be released in the future). From the LAKSIT dataset, we selected 3 image sequences composed of 3120 images each with a resolution of 1280×1024 pixels. We compare our system also with DynaSLAM Bescos et al. (2018), which is based on ORB-SLAM. The monocular ORB-SLAM is not always able to initialize on such sequences, because the apparent relative motion is not always consistent or the image is too blurred. Therefore, in

order to obtain a higher initialization rate and in order to demonstrate that 3D map-points are generated from keypoints which belong to fishes, we decreased the amount of keypoints required for the initialization step to 25. Since ORB-SLAM enters the *relocalization mode* only after the tracking is lost (and a minimum of 5 keyframes have been generated), several different sub-sequences have been chosen from the sequences, in order to find the longest drift accumulated in each sequence (see Fig. 5). The results are presented in Table 2.

With this keypoint rejection system, the ORB-SLAM system does not have to (re-)initialize, and drift does not accumulate over time, as most of the features detected at the moving objects would not be considered for mapping and tracking.
Approaches aiming to perform keypoint rejection for visual ego-motion estimation (like the one presented in this paper) using only *a priori* knowledge, may also reject *suitable* keypoints. Examples of this behaviour are shown in Fig. 6. We assume that in the underwater natural environment it is unlikely that *a priori* dynamic objects will remain static, and for this reason our approach is negligible impaired by this effect.

DynaSLAM provides two different rejection systems for the keypoints: a geometric test and a deep learning based method. While the geometric test is ineffective when elements in the scene are moving in a coherent way, the deep learning based method, which masks regions that should belong to moving objects, is effective in reducing the drift, even if the network is not trained for underwater scenarios.
Our system was also tested using the RT MVO dataset Ferrera et al. (2019) without benchmarking (unavailability of open source support). DynaSLAM as well as ORB-SlAM were tested with our proposed keypoint rejection system on the RT MVO data set and a similar performance for both systems with respect to the estimated trajectories were obtained. This is because the RT MVO sequences does show a very clear sea bed and the moving objects present in the sequences are only very small fishes, not showing for example any *schooling behaviour*.

Seq.	ORB-SLAM	DynaSLAM	DynaSLAM-M	Ours
1	29.13m	45.57m	7.75m	0m
2	11.24m	0m	0m	0m
3	14.68m	22.32m	3.09m	0m

Table 2. Absolute translation error (ATE) accumulated by ORB-SLAM with respect to three sequences out of the LAKSIT dataset, without and with the keypoint rejection system described in this paper (norm of the translation vector between the two initialization frames set to 1). ATE of 0m means that no initialization did occur.

The next experiments show the result of the different execution configurations with respect to floating point precision, parallel instances of the network and batch inference (we refer to Table 3). Note, due to the incompatibility to the Tensorflow *frozen model* with a multi-thread session creation procedure, no test has been performed with a parallel implementation running half precision floating point models.

Table 3 shows the results of our network with respect to execution speed using a desktop PC featuring 64Gb of RAM, a Nvidia GeForce RTX 2080 Ti and an Intel Core i9-9900K 8 Core @ 3.6Ghz, using a Python 3 implementation. The results are evaluated over a set of 5426 patches extracted from 10 test images. The various configurations evaluated differ by the type of processing unit utilized (CPU or GPU), the floating-point precision of the model, the number of threads launched (each executing a different instance of the graph; each instance elaborates an equal amount of patches), and whether we pre-loaded all the test patches in a single feeding dictionary. Regarding the floating-point precision, we tested the standard model created by Tensorflow, which uses 32 bit floating-point values both for constants and variables in the graph against an optimized version using 16 bit floating-points generated using the Nvidia TensorRT library. The values of mean and standard deviation in the table are computed over the entire images, averaging 543 keypoints (thus patches) each.

In Table 4 we compare the inference speed of DynaSLAM (Mask R-CNN) and the network used in this paper. The mean inference time has been obtained averaging over the time that took DynaSLAM to generate 100 masks from underwater images. Our approach is eight times faster on CPU and almost four times faster on GPU.

Our experiments showed that, given our network architecture and hardware, it is possible to achieve the highest inference speed with FP32 operations and 32 parallel network inferences on the GPU, performing at around 16 FPS.

5. CONCLUSION AND FURTHER WORK

In this paper a keypoint classification system is used to improve the robustness of visual ego-motion estimation in underwater environments. The procedure involved uses a deep convolutional neural network in order to classify each keypoint exploiting the surrounding image patch information. Only the keypoints which are classified as *suitable* for tracking will be retained for motion estimation.

The approach has been verified by comparing the drift accumulated by ORB-SLAM with a static camera recording a dynamic image scene of fishes showing a *schooling behaviour* with and without the keypoint rejection system described in this paper. The approach has been verified and benchmarked also against DynaSLAM, a state-of-the-art monocular visual SLAM system which accounts for dynamic environments.

We also discussed the inference performance for real-time operation by using state-of-the-art optimization frameworks for deep neural networks and parallel inference on CPU and GPU. The results show that the best performance can be achieved with a GPU acceleration, a floating point 32 network with 32 parallel sessions, even if the performance flattens out already under 16 parallel sessions. The method proposed in this paper enables feature-based underwater visual ego-motion estimation systems to oper-

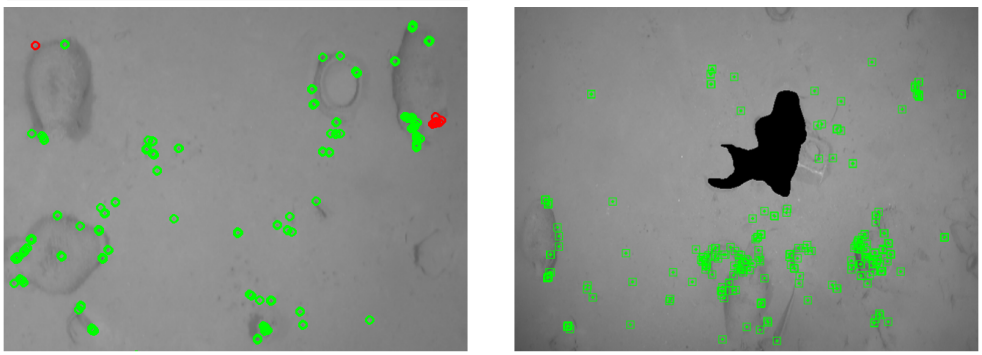

Fig. 6. This image shows the intrinsic risk of using the keypoint/area labeling algorithm: *suitable* keypoints/areas can be labeled as *unsuitable*, lowering the overall robustness of the visual motion estimation. Two different frames, coming from Sequence 1 of the RT MVO dataset are shown. On the left, circled in red, keypoints wrongly labeled as *unsuitable* from the network described in this paper are shown. On the right a masked area which did contain *suitable* seabed keypoints from DynaSLAM (Mask R-CNN) are shown.

Processing Unit	Mean	Std. Dev.	FP Precision	Threads	Batch
Graphic	0.134s	0.065s	32	1	✗
Graphic	0.080s	0.041s	32	4	✗
Graphic	0.067s	0.034s	32	16	✗
Graphic	**0.062s**	**0.031s**	32	32	✗
Graphic	0.066s	0.032s	32	64	✗
Graphic	1.139s	0.568s	32	1	✓
Graphic	1.139s	0.314s	*16*	1	✓
Central	0.377s	0.199s	32	1	✗
Central	0.413s	0.213s	*16*	1	✗
Central	0.213s	0.110s	32	8	✗
Central	0.201s	0.097s	32	16	✗
Central	3.401s	1.941s	32	1	✓
Central	3.359s	1.667s	*16*	1	✓

Table 3. Results of different network configurations for an inference speed analysis. *Batch* denotes that the inference is performed on all the patches in a single session.

Algorithm	Processing unit	Mean	Std. Dev.
Mask R-CNN	Central	1.665s	0.147s
This paper	Central	**0.201s**	0.167s
Mask R-CNN	Graphic	0.230s	0.284s
This paper	Graphic	**0.062s**	0.188s

Table 4. Inference speed of Mask R-CNN (the network present in DynaSLAM, which performs state-of-the-art object instance segmentation) versus the network present in this paper. The reported mean inference time corresponds to the best performing parallel implementation for both networks.

ate at close distance from regions of interest for underwater robotic applications, such as shipwreck or barrier reef monitoring, which are rich in marine life (causing dynamic image scenes). The proposed approach can be improved by using more training data representing unknown fish species, rock types, or other objects our CNN was not trained on. Data augmentation techniques, such as rotation, contrast manipulation and noise injection are likely to improve the generalization capabilities of the network. Therefore, it is advisable to train the network in such a manner that the false negative rate (i.e. the *suitable* keypoints that get classified as *unsuitable* keypoints) is as low as possible. Future work can also include optimizing the network for inference speed on more robotics-oriented computing platforms like the Nvidia Jetson AGX Xavier Nvidia (2019a) or even the Nvidia Jetson Nano Nvidia (2019b). Furthermore, the problem can be approximated by grouping neighbor keypoints and so reducing the amount of patches to be evaluated for each frame.

6. ACKNOWLEDGEMENTS

We would like to thank Christian Schellewald and Martin Føre for providing the three images sequences belonging to the LAKSIT project, the Australian Centre for Field Robotics, especially the marine robotics group for providing the *Scott Reef 25* and the *Tasmania O'Hara 7* dataset, the Institute for Marine and Antarctic Studies, University of Tasmania, for providing the *Tasmania Coral Point Count* and Jesse Eickholt, from the Central Michigan University, for providing the *An Underwater Observation Dataset of Fish*. This work was supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 - NTNU AMOS.

REFERENCES

Bescos, B., Fácil, J.M., Civera, J., and Neira, J. (2018). Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4), 4076–4083.

Bonin-Font, F., Oliver, G., Wirth, S., Massot, M., Negre, P.L., and Beltran, J.P. (2015). Visual sensing for autonomous underwater exploration and intervention tasks. *Ocean Engineering*, 93, 25–44.

Cui, L. and Wen, F. (2019). A monocular orb-slam in dynamic environments. In *Journal of Physics: Conference Series*, volume 1168, 052037. IOP Publishing.

Eickholt, J. (2018). An underwater observation dataset of fish. https://osf.io/3pyud/. Accessed: 2019-04-01.

Ferrera, M., Moras, J., Trouvé-Peloux, P., and Creuze, V. (2019). Real-time monocular visual odometry for turbid and dynamic underwater environments. *Sensors*, 19(3), 687.

Ferrera, M., Moras, J., Trouvé-Peloux, P., Creuze, V., and Dégez, D. (2018). The aqualoc dataset: Towards real-time underwater localization from a visual-inertial-pressure acquisition system. *arXiv preprint arXiv:1809.07076*.

for Field Robotics, A.C. (2009). Marine robotics datasets. http://marine.acfr.usyd.edu.au/datasets/. Accessed: 2019-06-01.

Føre, M., Frank, K., Svendsen, E., Schellewald, C., Sunde, L.M., Alfredsen, J.A. and Stahl, A. (2017). Final report on the project "Teknologi for nye datatyper og informasjon som beskriver situasjon og tilstand hos laksefisk i kommersielle merder (LAKSIT)" FHF Pnr. 901184. SINTEF report OC2017 A-104. Accessed: 2019-05-15.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Kim, A. and Eustice, R. (2009). Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1559–1565. IEEE.

Kim, A. and Eustice, R.M. (2013). Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29(3), 719–733.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krys, D. and Najjaran, H. (2007). Development of visual simultaneous localization and mapping (vslam) for a pipe inspection robot. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, 344–349. IEEE.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Leonardi, M., Stahl, A., Gazzea, M., Ludvigsen, M., Rist-Christensen, I., and Nornes, S.M. (2017). Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles. In *OCEANS 2017-Aberdeen*, 1–10. IEEE.

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

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

Nvidia (2019a). Jetson agx xavier. https://developer.nvidia.com/embedded/buy/jetson-agx-xavier. Accessed: 2019-06-01.

Nvidia (2019b). Jetson nano. `https://developer.nvidia.com/embedded/buy/jetson-agx-xavier-devkit`. Accessed: 2019-06-01.

OByrne, M., Pakrashi, V., Schoefs, F., and Ghosh, B. (2018). Semantic segmentation of underwater imagery using deep networks trained on synthetic imagery. *Journal of Marine Science and Engineering*, 6(3), 93.

Riazuelo, L., Montano, L., and Montiel, J. (2017). Semantic visual slam in populated environments. In *2017 European Conference on Mobile Robots (ECMR)*, 1–7. IEEE.

Richard, M.D. and Lippmann, R.P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4), 461–483.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf.

Tan, W., Liu, H., Dong, Z., Zhang, G., and Bao, H. (2013). Robust monocular slam in dynamic environments. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 209–218. IEEE.

Wangsiripitak, S. and Murray, D.W. (2009). Avoiding moving outliers in visual slam by tracking moving objects. In *2009 IEEE international conference on robotics and automation*, 375–380. IEEE.

6.4 Paper D: UVS: Underwater Visual SLAM - A Robust Monocular Visual SLAM System for Lifelong Underwater Operations

This paper is not yet published and is therefore not included.

Chapter 7

Additional Documents

7.1 UVS on Jetson Nano

UVS on Jetson Nano

1. INTRODUCTION

In this document, a set of tests are performed to understand if it's possible to run Underwater Visual SLAM (UVS) algorithm on an Nvidia Jetson platform and what are compromises to be made. First is presented a table containing the main hardware information about several Nvidia Jetson platforms, Table 1 (note that the form factor represents the dimension of the chip alone, not of the development board), then the performance analysis is discussed in the section that follows.

To perform the analysis the most demanding underwater visual sequence known to the author of this document is used, it is the Aqualoc dataset Ferrera et al. (2018) sequence 05. The Aqualoc dataset represents one of the first forms of publicly available dataset specifically targeting underwater visual SLAM. The two key parameters which influence UVS performance are the number of keypoints extracted during the keypoint extraction process, which happens for each frame, and the number of keyframes that are involved in the creation of the new map points, which happens according to a series of SLAM parameters, normally every 5 to 20 frames. All the combinations of parameters tested in this document successfully produce a SLAM estimate on all the sequences of the Aqualoc dataset, without loss of tracking.

2. RESULTS AND CONCLUSIONS

The results here presented are obtained using both the CPU and the GPU present on the Jetson Nano. The performance analysis present in Table 2 shows that UVS is not able to run when the power mode is set to 5W. Allowing the Jetson Nano to draw 10W, increases the performance by almost 4X. Disabling the visualization part of the system and disabling the Graphic User Interface (GUI) of the operative system further increase the performance, to the point of almost matching the 16 fps at which the sequence was captured. A bottleneck is represented by the storage system, which is an SD card. Performing pre-fetching allows for avoiding this bottleneck. With pre-fetching, it's possible to observe that UVS achieves the target fps and even passes it when the sequence framerate is artificially increased. The results obtained with pre-fetching are close to what to expect in a real implementation, assuming that the images are fed to UVS using an interface which does not require a lot of CPU cycles. Examples of methods for grabbing images using a small number of CPU cycles are the Camera Serial Interface (CSI) present on the Jetson and on the Raspberry Pi or USB cameras.

REFERENCES

Ferrera, M., Moras, J., Trouvé-Peloux, P., Creuze, V., and Dégez, D. (2018). The aqualoc dataset: Towards real-time underwater localization from a visual-inertial-pressure acquisition system. *arXiv preprint arXiv:1809.07076*.

Jetson	CPU	GPU	TFLOPs	Memory	Form factor	Power	Cost
Nano	4 core A57 @ 1.43 GHz	128-core Maxwell @ 921 MHz	0.472	4GB LPDDR4	45x70mm	5W-10W	99$
TX1	4 core A57 @ 1.73 GHz	256-core Maxwell @ 998 MHz	1	4GB LPDDR4	87mmx50mm	10W	✗ Not available
TX2/TX2i	4-core ARM Cortex-A57 @ 2 GHz, 2-core Denver2 @ 2 GHz	256-core Pascal @ 1.3 GHz	1.33	4GB/8GB LPDDR4	87mmx50mm	7.5W-15W	450$
Xavier NX	6-core NVIDIA Carmel @ 1.9 Ghz	384-core Volta @ 1.1Ghz , 48 Tensor cores	21	8GB LPDDR4	69.6mmx45mm	10W-15W	399$
AGX Xavier	8-core NVIDIA Carmel @ 2.26 GHz	512-core Volta @ 1.37 GHz, 64 Tensor core	32	16GB LPDDR4	100mmx87mm	10W-15W-30W	999$

Table 1. The Jetson family, GPU-centric System-On-Chips (SOCs).

Sequence	OS Gui	Visualization	Pre-fetching	Watt	Keypoints	Keyframes Matching	Targ. fps	Med. fps	Avg fps
Aqualoc 05	✓	✓	✗	10W	2000	20	16	15.25	7.24
Aqualoc 05	✓	✓	✗	10W	2000	5	16	11.40	7.59
Aqualoc 05	✓	✓	✗	10W	1500	20	16	13.21	6.87
Aqualoc 05	✓	✓	✗	10W	1500	5	16	22.67	7.89
Aqualoc 05	✓	✓	✗	5W	2000	20	16	4.02	2.57
Aqualoc 05	✓	✓	✗	5W	2000	5	16	4.12	2.85
Aqualoc 05	✓	✓	✗	5W	1500	20	16	4.77	3.1
Aqualoc 05	✓	✓	✗	5W	1500	5	16	5.14	3.8
Aqualoc 05	✓	✗	✗	10W	1500	5	16	14.87	10.72
Aqualoc 05	✗	✗	✗	10W	1500	5	16	22.8	13.2
Aqualoc 05	✗	✗	✗	10W	1500	5	**20**	19.71	**14.7**
Aqualoc 05	✗	✗	✓	10W	1500	5	16	22.14	**15.9**
Aqualoc 05	✗	✗	✓	10W	1500	5	20	23.8	**17.1**

Table 2. This table shows how UVS runs on the Jetson Nano, changing the power configuration and key parameters which more influence the performances.

7.2 Eelume Snake's Camera Sequences

Eelume Snake's Camera Sequences

Marco Leonardi

August 2020

1 Introduction

In this document, the performance of a specialized visual ego-motion estimation is analyzed with the help of two sequences captured from a single camera, part of Eelume's snake robot. The first sequence will be called the Garage's sequence and the second one will be called the TBS's sequence.
The images of the two supplied sequences show a significant **lossy compression noise** due to the format of the video file (MPEG-2), **this, unfortunately, compromises severely any visual ego-motion estimation system**, see Fig. 1. Unfortunately, no form of camera calibration is provided, so a pinhole camera model is used and the parameters are guessed.

2 Garage's sequence

In this sequence, the robot goes out of its garage, approaches a platform placed on the seabed which has some markers on it, then looks at ice blocks floating on the sea surface and goes back into its garage.

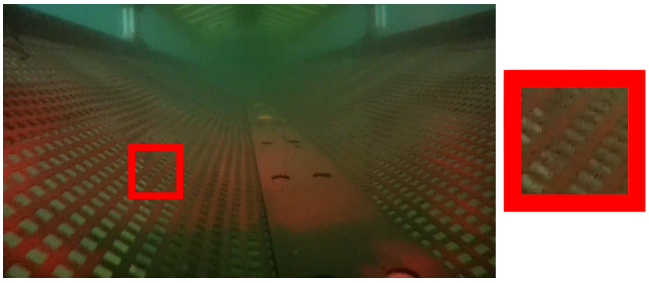

Figure 1: Motion JPEG artifacts in the Garage's sequence, showing how corners are heavily disrupted by this kind of lossy compression

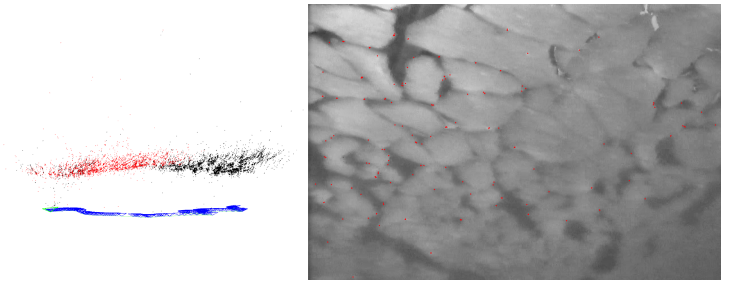

Figure 2: VSLAM under the ice block surface: the blocks results detailed enough for features to be correctly found and matched

2.1 Garage

This first part of the sequence is very challenging for visual ego-motion estimation systems, as the most gradient-rich areas are composed of repetitive patterns (the net), especially with feature-based methods.
While direct-based methods could perform better on repetitive patterns, they have been found to perform much worse in every single other underwater scenario, as they require accurate calibration and modeling of all the non-linearity involved.
A solution to provide visual-based positioning, in this case, is to use visual markers, for example, ArUco markers [1], like the ones present in this sequence around minute 7, but smaller, to fit the location. The markers could be covered with transparent anti-biofouling paint and/or subject to ultrasonic biofouling prevention [3], to keep required maintenance at the minimum. While biofouling would inevitably affect the markers in the long term, also the garage would be affected, the garage itself would eventually become visually diversified enough to turn itself into a VSLAM-friendly surface.

2.2 Platform on the seabed

The platform on the seabed contains four markers, unfortunately, they cannot be easily used for localization without access to the dictionary that was used to generate the markers. The generous size of the markers should allow a visual system to recognize them from a distance of several meters if turbidity allows.

2.3 Floating ice

Surprisingly the system is able to sporadically initialize and track while looking at ice blocks, see Fig. 2.

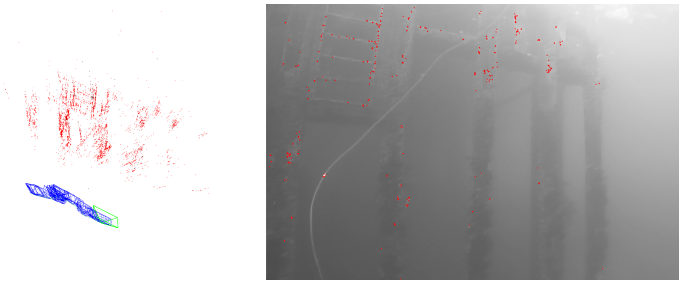

Figure 3: Pillars at TBS, in this picture, are shown the limits where features can be found and matched. In the top-left part of the image features are found, matched and correctly triangulated, while in the bottom-right part of the image the lack of gradients is too big for features to be found and correctly matched, the reasons can be found in the increasing observation distance and lack of illumination

3 TBS's Sequence

The sequence starts with the robot looking at the sea surface while submerged, then the garage appears in the scene. In the following part of the sequence, the robot navigates back and forth in front of a series of pillars. Several times the pillars exit temporarily from the scene. The final part of the sequence interest the seabed near the pillars.

3.1 Garage

While initialization, when the camera looks at the garage, is possible and the tracking is successful most of the time, the repetitive patterns which appear given the structure of the garage pose a threat to the system, similarly as in the previous sequence.

3.2 Pillars

Even with a lot of marine snow, the tracking when the camera moves around the pillars results in strong navigation, and obstacle avoidance solely on visual information would be possible, see 3.

3.3 Seabed

Tracking when the camera looks at the seabed result strong, with only minor surface deformation, due to incorrect calibration. In Fig. 4 it's possible to see the SLAM estimates just before the tracking is lost: the camera rises from the seabed and no structures are present in the scene, so the tracking is lost.

Sporadically, due to the presence of marine snow, the tracking continues instead of being lost, this is because the marine snow could create a *coherent set of correspondences*, see Fig. 5. Tracking of marine snow could be eliminated by visual-inertial fusion or through a neural network [2], at a cost of additional computing power.

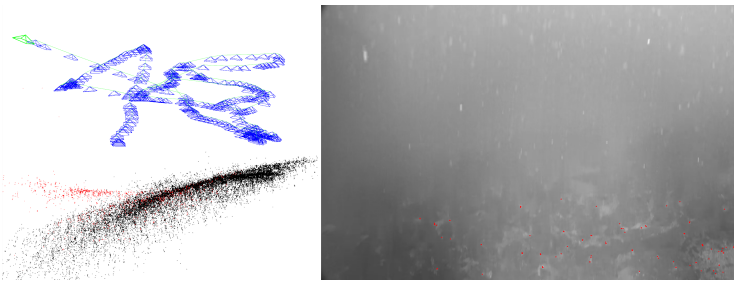

Figure 4: In this figure the SLAM estimates of a sequence where the camera (the frustum in green) after observing the seabed, starts to rise until no seabed can be seen anymore, and so the tracking is lost

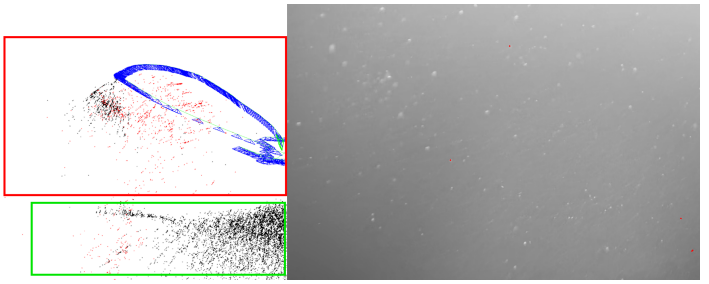

Figure 5: The undesirable tracking on marine snow: the lack of INS data combined with the tracking initialized on the sea bed and the lack of seabed in the scene tricks the system into thinking that moving marine snow represents the ego-motion of the camera. Inside the red rectangle there is the result of undesirable SLAM estimates on marine snow, while inside the green rectangle there is the previously estimated seabed

4 Conclusions

The sequences provided are heavily compressed with lossy compression, unfortunately, unsuitable for performing VSLAM, nevertheless, my work was able to provide SLAM estimates in all the key areas where it could be expected to perform so. The provided active illumination is sufficient, even if seems to be generated co-axially with the camera pinhole, this maximizes back-scattering, an undesired phenomenon. Camera calibration performed underwater would be beneficial, together with access to the raw images, to improve the performance in terms of quality and quantity of the SLAM estimates.

References

[1] Andrej Babinec, Ladislav Jurišica, Peter Hubinskỳ, and František Duchoň. Visual localization of mobile robot using artificial markers. *Procedia Engineering*, 96:1–9, 2014.

[2] Marco Leonardi et al. Deep learning based keypoint rejection system for underwater visual ego-motion estimation. https://app.cristin.no/results/show.jsf?id=1815961, 2020. Accessed: 2020-08-20.

[3] Ji-Soo Park and Jeung-Hoon Lee. Sea-trial verification of ultrasonic antifouling control. *Biofouling*, 34(1):98–110, 2018.

References

[1] a NTNU SINTEF OceanLab-project. The Applied Underwater Robotics Laboratory. https://www.ntnu.edu/aur-lab, 2022. [Online; accessed 06-November-2022].

[2] S. Agarwal, K. Mierle, and T. C. S. Team. Ceres Solver, 3 2022. URL https://github.com/ceres-solver/ceres-solver.

[3] A. Agrawal, S. Ramalingam, Y. Taguchi, and V. Chari. A theory of multi-layer flat refractive geometry. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3346–3353. IEEE, 2012.

[4] P. F. Alcantarilla and T. Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34 (7):1281–1298, 2011.

[5] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In *European conference on computer vision*, pages 214–227. Springer, 2012.

[6] A. Alcocer, P. Oliveira, and A. Pascoal. Underwater acoustic positioning systems based on buoys with gps. In *Proceedings of the Eighth European Conference on Underwater Acoustics*, volume 8, pages 1–8, 2006.

[7] G. Allibert, M.-D. Hua, S. Krupínski, and T. Hamel. Pipeline following by visual servoing for autonomous underwater vehicles. *Control Engineering Practice*, 82:151–160, 2019.

[8] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE transactions on robotics*, 24(5):1027–1037, 2008.

[9] A. Asada, F. Maeda, K. Kuramoto, Y. Kawashima, M. Nanri, and K. Hantani. Advanced surveillance technology for underwater security sonar systems. In *Oceans 2007-Europe*, pages 1–5. IEEE, 2007.

[10] A. Avetisyan. Fast binary descriptor matcher. https://github.com/skanti/HammingBruteForce, 2020. Accessed: 2020-01-01.

[11] H. Baek, B.-H. Jun, and M. D. Noh. The application of sector-scanning sonar: strategy for efficient and precise sector-scanning using freedom of underwater walking robot in shallow water. *Sensors*, 20(13):3654, 2020.

[12] A. S. Bandeira. A note on probably certifiably correct algorithms. *Comptes Rendus Mathematique*, 354(3):329–333, 2016.

[13] Y. Bar-Shalom and X.-R. Li. Estimation and tracking: Principles, techniques, and software [reviews and abstracts]. *IEEE Antennas and Propagation Magazine*, 38(1):62, 1996.

[14] D. Batra, B. Nabbe, and M. Hebert. An alternative formulation for five point relative pose problem. In *2007 IEEE Workshop on Motion and Video Computing (WMVC'07)*, pages 21–21. IEEE, 2007.

[15] G. Ben-Artzi, T. Halperin, M. Werman, and S. Peleg. Two points fundamental matrix. *arXiv preprint arXiv:1604.04848*, 2016.

[16] S. L. Bernstein, M. L. Burrows, J. E. Evans, A. Griffiths, D. McNeill, C. Niessen, I. Richer, D. White, and D. Willim. Long-range communications at extremely low frequencies. *Proceedings of the IEEE*, 62(3):292–312, 1974.

[17] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3 (4):4076–4083, 2018.

[18] F. Bonin-Font and A. Burguera Burguera. Nethaloc: A learned global image descriptor for loop closing in underwater visual slam. *Expert Systems*, 38(2): e12635, 2021.

[19] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[20] P. N. Brown and Y. Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3): 450–481, 1990.

[21] H. M. S. Bruno and E. L. Colombini. Lift-slam: A deep-learning feature-based monocular visual slam method. *Neurocomputing*, 455:97–110, 2021.

[22] A. Burguera and F. Bonin-Font. An unsupervised neural network for loop detection in underwater visual slam. *Journal of Intelligent & Robotic Systems*, 100(3):1157–1177, 2020.

[23] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[24] C. Campos, J. M. Montiel, and J. D. Tardós. Fast and robust initialization for visual-inertial slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1288–1294. IEEE, 2019.

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

[26] M. Caresta and N. Kessissoglou. Acoustic signature of a submarine hull. In *Fourteenth International Congress on Sound and Vibration (ICSV14)*, 2007.

[27] P. L. N. Carrasco, F. Bonin-Font, and G. O. Codina. Stereo graph-slam for autonomous underwater vehicles. In *Intelligent Autonomous Systems 13*, pages 351–360. Springer, 2016.

[28] M. Y. Cheung, D. Fourie, N. R. Rypkema, P. V. Teixeira, H. Schmidt, and J. Leonard. Non-gaussian slam utilizing synthetic aperture sonar. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3457–3463. IEEE, 2019.

[29] C. Cimarelli, H. Bavle, J. L. Sanchez-Lopez, and H. Voos. Raum-vo: Rotational adjusted unsupervised monocular visual odometry. *Sensors*, 22(7): 2651, 2022.

[30] M. Cummins and P. Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[31] E. Davies and M. Turk. *Advanced methods and deep learning in computer vision*. Elsevier, 2021.

[32] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. A column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 30(3):353–376, 2004.

[33] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[34] F. Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[35] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[36] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[37] P. Drews, E. Hernández, A. Elfes, E. R. Nascimento, and M. Campos. Real-time monocular obstacle avoidance using underwater dark channel prior. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4672–4677. IEEE, 2016.

[38] R. Edgar. Introduction to synthetic aperture sonar. *Sonar Systems*, pages 1–11, 2011.

[39] Eelume. Eelume Subsea Intervention. https://www.eelume.com, 2022. [Online; accessed 06-November-2022].

[40] R. Eliav and I. Klein. Ins/partial dvl measurements fusion with correlated process and measurement noise. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 4, page 34, 2018.

[41] R. Elvira, J. D. Tardós, and J. Montiel. Orbslam-atlas: a robust and accurate multi-map system. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6253–6259. IEEE, 2019.

[42] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[43] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*, 2016.

[44] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[45] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[46] R. M. Eustice. *Large-area visually augmented navigation for autonomous underwater vehicles*. PhD thesis, Massachusetts Institute of Technology, 2005.

[47] Z. Fang, D. Jiang, J. Huang, C. Cheng, Q. Sha, B. He, and G. Li. Autonomous underwater vehicle formation control and obstacle avoidance using multi-agent generative adversarial imitation learning. *Ocean Engineering*, 262:112182, 2022.

[48] F. Ferreira, G. Veruggio, M. Caccia, and G. Bruzzone. Real-time optical slam-based mosaicking for unmanned underwater vehicles. *Intelligent Service Robotics*, 5(1):55–71, 2012.

[49] M. Ferrera, J. Moras, P. Trouvé-Peloux, V. Creuze, and D. Dégez. The aqualoc dataset: Towards real-time underwater localization from a visual-inertial-pressure acquisition system. *arXiv preprint arXiv:1809.07076*, 2018.

[50] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[51] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[52] U. Frese. A proof for the approximate sparsity of slam information matrices. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 329–335. IEEE, 2005.

[53] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[54] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[55] J. Gao, X. An, A. Proctor, and C. Bradley. Sliding mode adaptive neural network control for hybrid visual servoing of underwater vehicles. *Ocean Engineering*, 142:666–675, 2017.

[56] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[57] R. Garcia and N. Gracias. Detection of interest points in turbid underwater images. In *OCEANS 2011 IEEE - Spain*, pages 1–9, 2011. doi: 10.1109/Oceans-Spain.2011.6003605.

[58] C. Gates. Antisubmarine warfare (asw) lexicon. Technical report, NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA, 1990.

[59] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[60] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. Openfabmap: An open source toolbox for appearance-based loop closure detection. In *2012 IEEE International Conference on Robotics and Automation*, pages 4730–4735. IEEE, 2012.

[61] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019.

[62] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez. Pl-svo: Semi-direct monocular visual odometry by combining points and line segments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4211–4216. IEEE, 2016.

[63] N. Gracias and J. Santos-Victor. Underwater video mosaics as visual navigation maps. *Computer Vision and Image Understanding*, 79(1):66–91, 2000.

[64] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–13, 2011.

[65] C. Gu, Y. Cong, and G. Sun. Environment driven underwater camera-imu calibration for monocular visual-inertial slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2405–2411. IEEE, 2019.

[66] R. E. Hansen, T. O. Sæbø, H. J. Callow, and P. E. Hagen. Interferometric synthetic aperture sonar in pipeline inspection. In *OCEANS'10 IEEE SYDNEY*, pages 1–10. IEEE, 2010.

[67] C. Harris, M. Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[68] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[69] R. I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10): 1036–1041, 1994.

[70] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593, 1997.

[71] M. Hildebrandt. *Development, Evaluation and Validation of a Stereo Camera Underwater SLAM Algorithm*. PhD thesis, Staats-und Universitätsbibliothek Bremen, 2014.

[72] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.

[73] J. Hong, K. de Langis, C. Wyethv, C. Walaszek, and J. Sattar. Semantically-aware strategies for stereo-visual robotic obstacle avoidance. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2450–2456. IEEE, 2021.

[74] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[75] A. Jurić, F. Kendeš, I. Marković, and I. Petrović. A comparison of graph optimization approaches for pose estimation in slam. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1113–1118. IEEE, 2021.

[76] H. A. Kadhim and W. A. Araheemah. A comparative between corner-detectors (harris, shi-tomasi & fast) in images noisy using non-local means filter. *Journal of Al-Qadisiyah for computer science and mathematics*, 11(3): Page–86, 2019.

[77] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[78] F. Kahl. Multiple view geometry and the l/sub/spl infin//-norm. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1002–1009. IEEE, 2005.

[79] R. E. Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82:35–45, 1960.

[80] v. W. C. Kebes at English Wikipedia, CC BY-SA 3.0. Electromagnetic absorption by water. https://commons.wikimedia.org/wiki/File:Absorption_spectrum_of_liquid_water.png, 2022. [Online; accessed 10-January-2022].

[81] A. Kim and R. M. Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29(3): 719–733, 2013.

[82] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[83] V. Kolmogorov, P. Monasse, and P. Tan. Kolmogorov and zabih's graph cuts stereo matching algorithm. *Image Processing On Line*, 4:220–251, 2014.

[84] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[86] P.-M. Lee, B.-H. Jeon, and S.-M. Kim. Visual servoing for underwater docking of an autonomous underwater vehicle with one camera. In *Oceans 2003. Celebrating the Past... Teaming Toward the Future (IEEE Cat. No. 03CH37492)*, volume 2, pages 677–682. IEEE, 2003.

[87] M. Leonardi and A. Stahl. Convolutional autoencoder aided loop closure detection for monocular slam. *IFAC-PapersOnLine*, 51(29):159–164, 2018.

[88] M. Leonardi and A. Stahl. Uvs: Underwater visual slam - a robust monocular visual slam system for lifelong underwater operations. *Autonomous Robots*, in press 2022.

[89] M. Leonardi, A. Stahl, M. Gazzea, M. Ludvigsen, I. Rist-Christensen, and S. M. Nornes. Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles. In *OCEANS 2017-Aberdeen*, pages 1–10. IEEE, 2017.

[90] M. Leonardi, L. Fiori, and A. Stahl. Deep learning based keypoint rejection system for underwater visual ego-motion estimation. *IFAC-PapersOnLine*, 53(2):9471–9477, 2020.

[91] E. Levitan and G. T. Herman. A maximum a posteriori probability expectation maximization algorithm for image reconstruction in emission tomography. *IEEE transactions on medical imaging*, 6(3):185–192, 1987.

[92] R. Li, S. Wang, Z. Long, and D. Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7286–7291. IEEE, 2018.

[93] R. Li, S. Wang, and D. Gu. Deepslam: A robust monocular slam system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68 (4):3577–3587, 2020.

[94] H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

[95] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[96] H. C. Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 227(1249):399–410, 1986.

[97] M. L. Lourakis and A. A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1526–1531. IEEE, 2005.

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

[99] T. Łuczyński, M. Pfingsthorn, and A. Birk. The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering*, 133:9–22, 2017.

[100] V. Lui and T. Drummond. An iterative 5-pt algorithm for fast and robust essential matrix estimation. In *BMVC*, 2013.

[101] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. doi: 10.1177/0278364916679498. URL http://dx.doi.org/10.1177/0278364916679498.

[102] E. Malis and M. Vargas. *Deeper understanding of the homography decomposition for vision-based control*. PhD thesis, INRIA, 2007.

[103] A. Martins, J. Almeida, C. Almeida, A. Dias, N. Dias, J. Aaltonen, A. Heininen, K. T. Koskinen, C. Rossi, S. Dominguez, et al. Ux 1 system design-a robotic system for underwater mining exploration. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1494–1500. IEEE, 2018.

[104] C. Mei, G. Sibley, and P. Newman. Closing loops without places. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3738–3744. IEEE, 2010.

[105] R. Miao, J. Qian, Y. Song, R. Ying, and P. Liu. Univio: Unified direct and feature-based underwater stereo visual-inertial odometry. *IEEE Transactions on Instrumentation and Measurement*, 2021.

[106] Y. Ming, X. Meng, C. Fan, and H. Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.

[107] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[108] D. Moreno, A. Burguera, and G. Oliver. Sss-slam: an object oriented matlab framework for underwater slam using side scan sonar. In *Proceedings of the XXXV Automation Conference*, 2014.

[109] M. Mousavi, S. Vaidya, R. Sutradhar, and A. Ashok. Openwaters: Photo-realistic simulations for underwater computer vision. In *Proceedings of the 15th International Conference on Underwater Networks & Systems*, pages 1–5, 2021.

[110] R. Mur-Artal and J. D. Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853. IEEE, 2014.

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

[112] R. Mur-Artal and J. D. Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

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

[114] P. L. Negre, F. Bonin-Font, and G. Oliver. Cluster-based loop closing detection for underwater slam in feature-poor regions. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2595. IEEE, 2016.

[115] e. a. Negre Carrasco. Global image signature for visual loop-closure detection. *Autonomous Robots*, 2016.

[116] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation*, 17 (6):890–897, 2001.

[117] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[118] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *IEEE Robotics and Automation Letters*, 3(3):2346–2353, 2018.

[119] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[120] S. M. Nornes, M. Ludvigsen, Ø. Ødegard, and A. J. SØrensen. Underwater photogrammetric mapping of an intact standing steel wreck with rov. *IFAC-PapersOnLine*, 48(2):206–211, 2015.

[121] U. of Hawaii at Manoa. The ocean is largely unexplored. https://manoa.hawaii.edu/exploringourfluidearth/standards-alignment/ocean-literacy-principles-olp/olp-7-ocean-largely-unexplored, 2021. [Online; accessed 03-October-2021].

[122] U. of Zurich. The ocean is largely unexplored. https://rpg.ifi.uzh.ch/svo_pro.html, 2021. [Online; accessed 02-Febraury-2022].

[123] M. Oladi, A. Ghazilou, S. Rouzbehani, N. Z. Polgardani, K. Kor, and H. Ershadifar. Photographic application of the coral health chart in turbid environments: The efficiency of image enhancement and restoration methods. *Journal of Experimental Marine Biology and Ecology*, 547:151676, 2022.

[124] P. Ozog, G. Troni, M. Kaess, R. M. Eustice, and M. Johnson-Roberson. Building 3d mosaics from an autonomous underwater vehicle, doppler velocity log, and 2d imaging sonar. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1137–1143. IEEE, 2015.

[125] V. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella. Gyroscope technology and applications: A review in the industrial perspective. *Sensors*, 17(10):2284, 2017.

[126] W. S. Pegau, D. Gray, and J. R. V. Zaneveld. Absorption and attenuation of visible and near-infrared light in water: dependence on temperature and salinity. *Applied optics*, 36(24):6035–6046, 1997.

[127] S. Pi, B. He, S. Zhang, R. Nian, Y. Shen, and T. Yan. Stereo visual slam system in underwater environment. In *OCEANS 2014-TAIPEI*, pages 1–5. IEEE, 2014.

[128] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE, 2014.

[129] R. M. Pope and E. S. Fry. Absorption spectrum (380–700 nm) of pure water. ii. integrating cavity measurements. *Appl. Opt.*, 36(33):8710–8723, Nov 1997. doi: 10.1364/AO.36.008710. URL http://www.osapublishing.org/ao/abstract.cfm?URI=ao-36-33-8710.

[130] M. J. Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, 1970.

[131] S. Rahman, A. Q. Li, and I. Rekleitis. Sonar visual inertial slam of underwater structures. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5190–5196. IEEE, 2018.

[132] R. Ranftl and V. Koltun. Deep fundamental matrix estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 284–299, 2018.

[133] D. Ribas, P. Ridao, J. D. Tardós, and J. Neira. Underwater slam in a marina environment. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1455–1460. IEEE, 2007.

[134] K. Richmond. *Real-time visual mosaicking and navigation on the seafloor*. Stanford University, 2009.

[135] B. M. H. Romeny. *Front-end vision and multi-scale image analysis: multi-scale computer vision theory and applications, written in mathematica*, volume 27. Springer Science & Business Media, 2008.

[136] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[137] P. L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[138] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

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

[140] J. Salvi, Y. Petillo, S. Thomas, and J. Aulinas. Visual slam for underwater vehicles using video velocity log and natural landmarks. In *OCEANS 2008*, pages 1–6. IEEE, 2008.

[141] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. *Advances in neural information processing systems*, 18, 2005.

[142] Y. Y. Schechner and N. Karpel. Clear underwater vision. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[143] M. She, Y. Song, J. Mohrmann, and K. Köser. Adjustment and calibration of dome port camera systems for underwater vision. In *German Conference on Pattern Recognition*, pages 79–92. Springer, 2019.

[144] J. Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

[145] K. Siantidis. Side scan sonar based onboard slam system for autonomous underwater vehicles. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 195–200. IEEE, 2016.

[146] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[147] H. Strasdat, J. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2(3):7, 2010.

[148] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. In *2011 international conference on computer vision*, pages 2352–2359. IEEE, 2011.

[149] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[150] A. Tal, I. Klein, and R. Katz. Inertial navigation system/doppler velocity log (ins/dvl) fusion with partial dvl measurements. *Sensors*, 17(2):415, 2017.

[151] C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

[152] B. Teixeira, H. Silva, A. Matos, and E. Silva. Deep learning for underwater visual odometry estimation. *IEEE Access*, 8:44687–44701, 2020.

[153] M. Trajković and M. Hedley. Fast corner detection. *Image and vision computing*, 16(2):75–87, 1998.

[154] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5038–5047, 2017.

[155] S. Urban, J. Leitloff, and S. Hinz. Mlpnp-a real-time maximum likelihood solution to the perspective-n-point problem. *arXiv preprint arXiv:1607.08112*, 2016.

[156] C. F. Van Loan and G. Golub. Matrix computations (johns hopkins studies in mathematical sciences). *Matrix Computations*, 1996.

[157] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[158] Y. Wang, X. Ma, J. Wang, S. Hou, J. Dai, D. Gu, and H. Wang. Robust auv visual loop-closure detection based on variational autoencoder network. *IEEE Transactions on Industrial Informatics*, 18(12):8829–8838, 2022.

[159] J. Weickert, B. T. H. Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE transactions on image processing*, 7(3):398–410, 1998.

[160] J. Weickert, S. Grewenig, C. Schroers, and A. Bruhn. Cyclic schemes for pde-based image analysis. *International Journal of Computer Vision*, 118 (3):275–299, 2016.

[161] N. Weidner, S. Rahman, A. Q. Li, and I. Rekleitis. Underwater cave mapping using stereo vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5709–5715. IEEE, 2017.

[162] R. Williams, A. J. Wright, E. Ashe, L. Blight, R. Bruintjes, R. Canessa, C. Clark, S. Cullis-Suzuki, D. Dakin, C. Erbe, et al. Impacts of anthropogenic noise on marine life: Publication patterns, new discoveries, and future directions in research and management. *Ocean & Coastal Management*, 115:17–24, 2015.

[163] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR 2011*, pages 3057–3064. IEEE, 2011.

[164] Y. Wu, X. Ta, R. Xiao, Y. Wei, D. An, and D. Li. Survey of underwater robot positioning navigation. *Applied Ocean Research*, 90:101845, 2019.

[165] C. Xu and B. Zheng. Underwater image feature matching based on inhomogeneous illumination. In *Advanced Technologies in Ad Hoc and Sensor Networks*, pages 379–385. Springer, 2014.

[166] X. Yang and K.-T. Cheng. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In *2012 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 49–57. IEEE, 2012.

[167] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European conference on computer vision*, pages 467–483. Springer, 2016.

[168] X. Yuan, J.-F. Martínez-Ortega, J. A. S. Fernández, and M. Eckert. Aekf-slam: A new algorithm for robotic underwater navigation. *Sensors*, 17(5): 1174, 2017.

[169] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[170] J. Zhao, W. Xu, and L. Kneip. A certifiably globally optimal solution to generalized essential matrix estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12034–12043, 2020.

[171] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017.

[172] M. Zuliani, C. Kenney, and B. Manjunath. A mathematical comparison of point detectors. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 172–172. IEEE, 2004.

[173] P. G. O. Zwilgmeyer, M. Yip, A. L. Teigen, R. Mester, and A. Stahl. The varos synthetic underwater data set: Towards realistic multi-sensor underwater data with ground truth. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3722–3730, October 2021.

ISBN 978-82-326-5517-5 (printed ver.)
ISBN 978-82-326-6310-1 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology