Autonomous navigation and mapping for underwater snake robots

TTK4551 - Engineering Cybernetics, Specialization Project
Department of Engineering Cybernetics
Norwegian University of Science and Technology

Andreas Våge

NTNU – Trondheim
Norwegian University of
Science and Technology

Supervsisor: Edmund Førland Brekke, Førsteamanuensis NTNU ITK
Co-supervisor: Professor Kristin Y. Pettersen, NTNU ITK
Professor Annette Stahl, NTNU ITK
Pål Liljebäck, CTO Eelume AS

Problem description

In all major subsea industries, such as oil and gas and aquaculture, there is currently a large and increasing demand for autonomous underwater vehicles (AUVs) to carry out various types of inspection and intervention tasks. Very often, the need for autonomy is based on a need for reducing the use of costly surface vessels required to carry out underwater operations with remotely operated vehicles (ROVs).

To achieve high levels of autonomy in underwater robots, it is essential that these robots can use machine vision, such as optical cameras or sonars, to map their environment and navigate based on this map. This capability is the focus of this project and requires that the robot can carry out SLAM (Simultaneous Localization and Mapping), as well as autonomous motion planning and navigation in the mapped environment. The project will be carried out in close collaboration with Eelume AS, a spin-off company from NTNU developing underwater snake robots in cooperation with Statoil and Kongsberg Maritime. The Eelume vehicles are designed to live in docking stations on the seabed over extended periods in order to carry out operations on subsea infrastructure (visual inspection, valve operations, etc) without the need for surface vessel support. Autonomous navigation and mapping is an essential capability for the upcoming autonomy of these vehicles.

The objective of this project is to implement underwater SLAM algorithms based on optical cameras, where an Eelume robot is considered as the target platform. The Eelume vehicle is a particularly relevant platform for SLAM since its long body allows for a good spatial distribution of the cameras and lights used for mapping, thereby

enabling cameras and lights from different angles. The overall objective is to enable the robot to develop a 3D map of its environment (such as the seabed, structures on the seabed, or floating structures) and also to enable the robot to continuously determine its own location as it moves in the mapped environment.

An additional objective is to develop algorithms where the vehicle autonomously plans and executes motion with respect to the mapped environment, such as:

- station keeping based on visual odometry (by fixating on features in the camera image).
- scanning around an object at a fixed distance.
- following a pipeline on the seabed.

An Eelume vehicle, as well as the lab basin of Eelume, will be made available to this project work for experimental testing. The use of a simulator currently being developed by NTNU and Eelume should also be considered for early testing.

As this task is intented to be worked on by three students it has been further devided into three main elements, SLAM, control and detection.

The report shall be written in English and edited as a research report including Abstract, Introduction with motivation, literature survey, contributions of the project work, and the outline of the report. This is followed by the chapters describing the results of the project work, simulation results and corresponding discussion, and a conclusion including a proposal for further work. It is assumed that the Department of Engineering Cybernetics, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work.

Utført ved Institutt for teknisk kybernetikk
Faglærer:

Edmund Førland Brekke, Førsteamanuensis NTNU ITK

Medveiledere:

Professor Kristin Y. Pettersen, NTNU ITK

Professor Annette Stahl, NTNU ITK

Pål Liljebäck, CTO Eelume AS

Abstract

This report presents the work done in preparation for implementation of an underwater visual SLAM algorithm in a underwater snake robot. In order to do good development and evaluation of such a system a acoustic underwater localisation system have been installed and tested in the pool at the Eelume lab. Further two stereo camera rigs have been assembled and calibrated using two different camera models. The best resulting system have been used to evaluate state-of-the-art visual SLAM algorithms in the Eelume pool.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This introductory chapter will briefly provide context for the material/results presented in this report and give motivation and a description of the problem to be solved. The scope of the work is then defined through a list of assumptions. Finally, the contributions of the thesis are defined and elaborated.

In all major subsea industries, such as oil & gas and aquaculture, there is currently a large and increasing demand for autonomous underwater vehicles (AUVs) to carry out various types of inspection and intervention tasks. Very often, the need for autonomy is based on a need for reducing the use of costly surface vessels required to carry out underwater operations with remotely operated vehicles (ROVs).

To achieve high levels of autonomy in underwater robots, it is essential that these robots can use machine vision, such as optical cameras or sonars, to map their environment and navigate based on this map. This capability is the focus of this project and requires that the robot can carry out SLAM (Simultaneous Localization and Mapping).

There will be a focus on visual SLAM in this project, that is SLAM methods based on optical cameras. The camera faces several challenges underwater:

- Turbidity.

- Unstructured environment: few edges and features.
- Light scattering, refraction and blurring.
- Visual degradations induced by the medium properties.

These effects cause the cameras to have limited range and make it difficult finding good geometric models and very difficult finding photometric models, (Corke et al.; 2013). However the camera provide a lot of information, and the ability to recognise places it has been before are very useful during mapping, (Agarwal et al.; 2011; Bryson et al.; 2013).

SLAM has gotten a huge boost lately due to the interest from autonomous car industry, cheap quad copters and VR and AR as well as cheaper/smaller computational power and cameras. Great work by the research community sharing open-cource code and detailed publications. All this has resulted in great visual SLAM system working well above water which are now entering industry (VR/AR/autonomous drones/self driving cars etc..).

However there have been less work for under water use, due to more difficulty and expensiveness of generating test data as well as less people are interested in it. But now, when the above water methods have stabilised, reached a satisfactory level of robustness and accuracy it is time to look under water . Even though the under water interest is less than above water , the need is much bigger. There are huge unexplored areas under water . There is no gps signal. Humans can not be under water . This is what robotics should be all about. This is the ideal usecase for robotics: Creating robots that can do things humans can not do. Good under water SLAM is the key component for truly autnoumous under water robots.

Why visualunder water SLAM: Cheap sensors, detailed mapping, perhaps the most complex sensor available. If the visualpart is understood adding additional sensor data such as SONAR would be easier as the information is much closer to the actual SLAM problem: It is already geometric information. While visualsensors provide photometric info which would have to be preprosessed using keypoints and descriptors and descriptor matching to get geometric data which then can be used in slam. (Or even more complicated use direct methods which do SLAM using the photometric info

directly.) So if we need visualdata in our problem, it would be reasonable to get that working first.

1.1 Assumptions

For this project it is assumed that the snake robots will operate in structured environments without large amounts of turbidity our other occlusions. For the project all test will be done in a indoor pool, but the snake-robots are planned to be used for inspection and maintenance, and thus it is naturally to assume they will operate in man-made environments in real scenarios as well. Further they are planned to be used at around 300 meters depth, where there are little fish and turbidity, thus these assumptions are not too unrealistic. It is also assumed that there are strong uniformed lightening covering the area the snake robot is operating. These are met in the pool, however it is uncertain if these will be met in real scenarios, but keep in mind the shape of the snake robot allow for good spacing between camera and the lights, thus it is possible these assumptions are met. Further it is assumed that the robot will move relatively smooth, without huge roll movements.

1.2 Background and Contributions

Contributions: Necessary preparations for implementing an under water visualSLAM method.

- Underwater acoustic positioning system setup and calibration, including creating ROS driver.
- Camera setup and calibration.
- Literature review
- Creation of under water dataset
- Evaluation of modern visualSLAM methods under water .

Except for the ROS driver for the underwater acoustic positioning system, I have not contributed to very much new source code. Most of the results are based on open source code, however there were quite some work fixing old dependencies and maintaining the source code.

- Camera setup:
  - ROS Point Grey camera. driver[1].
  - ROS OpenCV video streamer[2].
- Camera calibration:
  - ROS camera calibration[3]
  - Kalibr[4] (Oth et al.; 2013).
  - Pinax model[5] (Łuczyński et al.; 2017)
- SLAM methods:
  - ORB-SLAM2[6] (Mur-Artal and Tardos; 2016)

PhD candidate Marco Leonardi provided nice build guide for the Pinax model, including a fix to an old, very tricky broken build dependency. He also helped with a lot of the recordings for camera calibration and the final data sets. Master student Øystein Barth Utbjoe also helped with recording calibration data sets as well as setup and calibration of the underwater acoustic positioning system.

1.3 Outline

The report is organised as follows: In Chapter 2 we describe the the problem of optimisation based SLAM, including a brief probabilistic formulation. Then in chapter 3

[1] https://github.com/ros-drivers/pointgrey_camera_driver
[2] http://wiki.ros.org/video_stream_opencv
[3] https://github.com/ros-perception/image_pipeline
[4] https://github.com/ethz-asl/kalibr
[5] https://github.com/pinax/pinax-models
[6] https://github.com/raulmur/ORB_SLAM2

we go deeper into how some selected popular SLAM methods work and discusses which would be suitable to try underwater. In chapter 4 we go through the preparations and experiments conducted during the project, while in chapter 5 we conclude and discuss future work.

Chapter 2

SLAM

SLAM refers to the problem of estimating the trajectory of a robotic vechile over time while simultaneously estimating a map of the surrounding environment, Cadena et al. (2016).

SLAM or simultaneously localisation and mapping is a fundamental problem within robotics. It allow the robot to observe its environment with respect to itself, thus giving spatial awareness. We will focus on visual SLAM, where one uses cameras as sensors, but there are also SLAM systems for other sensors like lidar. In theory one could perform SLAM with any sensor suit where one is able to observe the environment and recognise places one has been before, so called place recognition. Topics similar to visual SLAM include structure from motion (SFM) and visual odometry (VO). Visual odometry is the problem of estimating the movement of a camera system purely by using the images from the cameras and is the buidling block of any visual SLAM method. Structure from motion is the general problem of generating 3D structures from 2D images. This is possible by looking at the structures from different views etc. by moving the camera. In order to generate the 3D structures SFM first needs to estimate the pose of the cameras when the 2D images where taken. Thus it solves the same problem as SLAM. However SFM generally do not run in real time, the most known example of SFM, "Building Rome in a Day", Agarwal et al. (2011), actually used

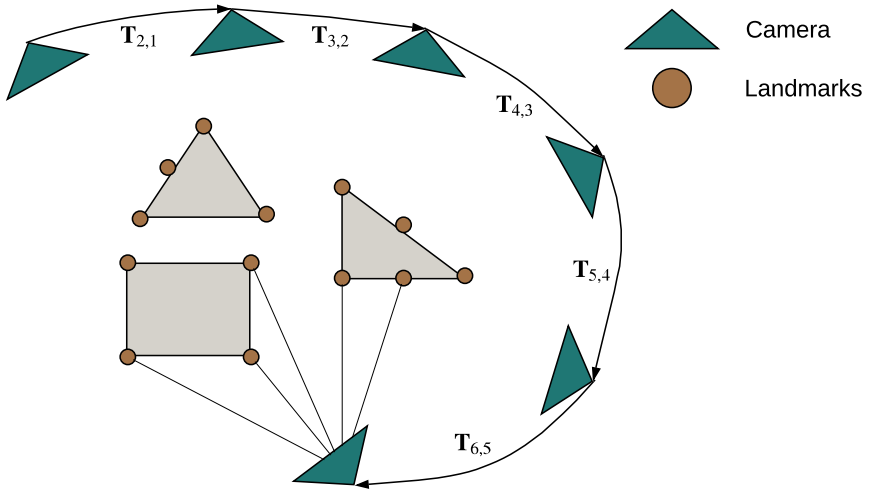


Figure 2.1: Simple 2D Visual Odometry: estimate the trajectory of the camera by tracking the landmarks in each image. Visual SLAM is an extension where one also estimate a global map of the landmarks and handles loop closures.

one day (21 hours) to complete its calculation using a cluster with 500 compute cores, still impressive considering it was 150K images, mostly taken by tourists. SLAM and VO needs to run in real time, allowing the robot to make decisions based the current time instead of the present.

The input to a SLAM algorithm is a stream of noisy measurements $Z = \{z_i : i = 1, \ldots, n_z\}$ from the real world and the objective is to estimate the state $\mathcal{X}$. Which consist of the state of the robot and the map of the environment.

$$\mathcal{X} = \begin{bmatrix} \eta \\ m \end{bmatrix} = \begin{bmatrix} \text{Robot state} \\ \text{Map} \end{bmatrix}$$

In visual SLAM, the robot state is normally just the ego-motion of the robot represented as a set of 3D poses, $\eta = \{x_i : i = 1, \ldots, n_x\}$, sampled from the robot

trajectory.

$$x_i = \mathbf{T}_{b_i,b_0} = \begin{bmatrix} \mathbf{R}_{b_i,b_0} & t_{b_i,b_0} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.1}$$

Here the pose is represented using homogeneous transformations matrices, $\mathbf{T} \in \mathbf{SE}(3)$ consisting of a rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$ and a translation vector $t \in \mathbb{R}^3$, but it could also be represented using other representations like quaternions or twists. The subscript b_i, b_0 represents time instances of the coordinate systems used. For visual SLAM b_i would be the camera coordinate system for the current pose and b_0 would typically be the first camera pose. From the visual odometry one get the pose of the camera relative to the last frame, see figure 2.1, then one can accumulate the measurements to get the position relative to the first frame.

$$\mathbf{T}_{b_i,b_0} = \mathbf{T}_{b_i,b_{i-1}}\mathbf{T}_{b_{i-1},b_0} \tag{2.2}$$

For more complete SLAM systems, perhaps with multiple sensors b_i could be the body frame at index i and b_0 could be the inertial frame I. The robot state could also include sensor information like the intrinsic parameters of the camera or, in case of visual-inertial SLAM, velocity of the robot and bias estimates for the IMU.

For visual SLAM the map is typically a set of landmarks $m = \{l_i : i = 1, ..., n_l\}$ represented as 3D points.

2.1 Probabilistic formulation

Like most estimation problems, the fundamental problem of SLAM is formulated in a probabilistic framework. One can think of probability theory as the bridge between the well-defined mathematics and the random real world. In a probabilistic framework the state is modelled as stochastic variable, $\mathcal{X}_{stoc}$, with a probability density function $p_{\mathcal{X}_{stoc}}(\mathcal{X})$. Such that the probability that the stochastic variable $\mathcal{X}_{stoc}$ is equal to some value $\mathcal{X}$ is $p_{\mathcal{X}_{stoc}}(\mathcal{X})$. This is written as $\Pr(\mathcal{X}_{stoc} = \mathcal{X}) = p_{\mathcal{X}_{stoc}}(\mathcal{X})$. We simplify the notation $p_{\mathcal{X}_{stoc}}(\mathcal{X})$ to $p(\mathcal{X})$ as is done in Cadena et al. (2016), which a lot of our

probabilistic notation is based on, and most of the literature.

The SLAM problem is usually formulated as a maximum a posteriori estimation problem, that is, we define the state estimator $\hat{\mathcal{X}}_{\text{MAP}}$ as the assignment of the state $\mathcal{X}_{stoc}$ that maximise the probability of obtaining the given measurements.

$$\hat{\mathcal{X}}_{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}|Z) \tag{2.3}$$

Using the Bayes theorem we can reformulate the problem

$$\begin{aligned}\hat{\mathcal{X}}_{\text{MAP}} &= \underset{\mathcal{X}}{\operatorname{argmax}}\, \frac{p(Z|\mathcal{X})p(\mathcal{X})}{p(Z)} \\ &= \underset{\mathcal{X}}{\operatorname{argmax}}\, p(Z|\mathcal{X})p(\mathcal{X})\end{aligned}$$

Where $p(Z|\mathcal{X})$ is the likelihood of the measurements Z given the state $\mathcal{X}$. $p(\mathcal{X})$ is a prior probability of the state. The last equality follow from the fact that $p(Z)$ is independent of $\mathcal{X}$.

Remark 2.1.1 *If we do not have any prior information or $p(\mathcal{X})$ is a uniform distribution, then* (2.3) *becomes equal to a maximum likelihood estimator.*

Assuming the measurements Z are independent, we can factorise (2.3) into

$$\hat{\mathcal{X}}_{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmax}}\; p(\mathcal{X}) \prod_{i=1}^{n_z} p(z_i|\mathcal{X}_i) \tag{2.4}$$

where $\mathcal{X}_i$ is all the states that z_i is dependent upon, for visual SLAM, where z_i is a image, $\mathcal{X}_i$ represents all the landmarks visible in that image as well as the pose of the camera when the image was taken.

Definition 2.1.1 (Factor graph, Kschischang et al. (2001)) *Suppose $g(x_1, \ldots, x_n)$ factors into a product of several local functions, f_j each having some subset X_j of $\{x_1, \ldots, x_n\}$ as arguments; i.e., suppose that*

$$g(x_1, \ldots, x_n) = \prod_{j \in J} f_j(X_j) \tag{2.5}$$

A factor graph is a bipartite graph that expresses the structure of the factorisation (2.5). *A factor graph has a variable node for each variable, a factor node for each local function, and an edge-connecting variable node, x_i to factor node f_j if and only if x_i is an argument of f_j.*

If we define a factor node ϕ_i as

$$\phi_i(X_i) = \begin{cases} p(z_i|\mathcal{X}_i), & \text{for } i \in \{1 \ldots n_z\} \\ \phi_{prior} = p(\mathcal{X}), & \text{for } i = 0 \end{cases} \tag{2.6}$$

we see that (2.4) can be written as

$$\hat{\mathcal{X}}_{MAP} = \underset{\mathcal{X}}{\operatorname{argmax}} \prod_{i=1}^{n_z} \phi_i(X_i) \tag{2.7}$$

which by definition 2.1.1, is a factor graph optimisation problem.

As introduced by Klein and Murray (2007) we separate between the backend and frontend of a SLAM method. The backend solves (2.7), while the frontend sets it up. That includes collecting all the necessary data, do data association and estimate an initial solution (Huang and Dissanayake; 2016).

If we had all the data available from the go, we could first run the frontend one time, then run the backend. This would be a typical SFM problem. Since a SLAM algorithm need to run in real time, new measurements arrive continuously and the frontend and backend need to interact and run continuously. There are several ways of implementing this interaction, we will look into how some of the most popular methods have done it. See for example figure 3.2.

2.2 Classification of SLAM methods

Often when talking about SLAM methods we classify them by two measures:

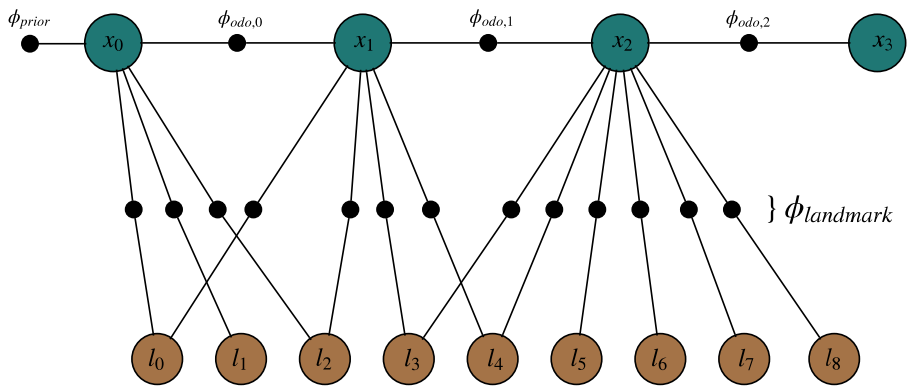


Figure 2.2: Factor graph example. Green circle represents the robot states, the brown circles are the map represented as landmarks. The small black circles represents the factors. The lines shows which variables each factor are depending on.

- Direct vs Indirect
- Sparse vs Dense

Sparse vs dense simply indicates how dense the resulting map is.

An direct SLAM/VO method optimises directly over the image intensities. Ideally, to determine how much the camera have moved since last frame, a direct method uses a complete model of the camera to warp the current image step by step until it looks exactly like the last frame. The camera pose and map representation that made the two images look exactly the same are then the solution. For direct methods the factor $\phi_i(X_i)$ becomes equal the photo-metric error. The photo-metric error can be thought upon as an error describing how similar two images are.

An indirect SLAM/VO method reduces the problem into a geometric problem by first using image features and descriptors. The first thing a indirect method does, to find out how much the camera have moved since last frame, is to extract image features and match them between the two frames. Then the reaming problem is similar to as a direct method. It uses a geometric camera model to tell which pose of the camera and 3D position of the map-points that makes the points in both the two images correspond

to the points in the map. For indirect methods the factor $\phi_i(X_i)$ becomes equal the reprojection error, and the optimisation problem (2.7) is often referred to as Bundle adjustment. The reprojection error is an error measuring the sum of distances between measured points in a image and the estimated 3D points reprojected back into the 2D image.

While a direct method ideally uses the entire image, all the information available, a indirect method looses a lot of the information when reducing the problem to a geometric model.

Chapter 3

Literature review

We will look more into the relevant literature and available open source SLAM methods, while trying to keep a focus on optimisation based visual SLAM. First a quick review of the reviews. Table 3.1 list some of the most relevant resent surveys both for under and above water.

Table 3.1: Relevant surveys

Year	Citation	Topic
2018	Chen, Zhu, Li and You	Visual-Inertial SLAM
2018	Delmerico and Scaramuzza	Visual-Inertial Odometry
2018	Saputra et al.	Visual SLAM in dynamic Environments
2017	Lu et al.	Underwater Optical Image Processing
2017	Wang, Zhang and An	SLAM on unstructured lunar environment
2016	Cadena et al.	Past present and future of SLAM
2016	Lowry et al.	Visual place recognition
2015	Hidalgo and Braunl	Underwater SLAM techniques
2015	Massot-Campos and Oliver-Codina	Underwater 3D reconstruction
2014	Paull et al.	AUV Navigation and Localisation

In the past most underwater navigation and localisation methods were based on

expensive inertial sensors or installed beacons in the region of interest, but with the advances of SONAR and optical SLAM there is a new future in sight where the limitations of pre-installed beacons can be surpassed Paull et al. (2014). Massot-Campos and Oliver-Codina (2015) performs a comprehensive classification of different sensors and techniques used for underwater 3D reconstruction. For many methods they even includes a estimate of accuracy and resolution. They estimate that stereo vision and SfM have less than 3 m range in typical seawater. Turbidity and occlusions like fish is a challenge. So far SLAM methods have typically been assuming a constant environment, but allowing for dynamic environments are in progress (Saputra et al.; 2018).

Lu et al. (2017) presents a review of the state-of-the art within underwater optical image processing techniques, for both hardware (Polarization, range-gate, stereo imaging etc..) and software techniques (Wavelenght compensation based on physical- and non physical- models as well as colour reconstruction). One interesting method by Gracias et al. (2008): they presented a solution to the shallow water flickering problem that arises because of the waves effect on the sunlight. They observed that neighbouring images from a video sequence differ in two ways, one is related to the illumination field and the other to the registration error, caused by imperfect image stitching. Using a low pass filter, they were able to separate the difference caused by flickering, then use it to correct the original image while maintaining the sharpness of the image. However they do not state their run time, and for every image the process includes extracting and matching robust features and computing the temporal median of several images, thus it could be challenging to incorporate it in a real time SLAM system. However it could be used for instance only over the keyframes.

3.1 Model-assisted bundle adjustment

Model-assisted bundle adjustment is an interesting topic within SLAM, particularry in our case where our snake robot will operate in desert areas at the bottom of the sea with only a few man-made structures. There will exist models of the man-made structures, and model-assisted bundle adjustment is about including these model into the optimisation problem. Early work on model assisted bundle adjustment used for

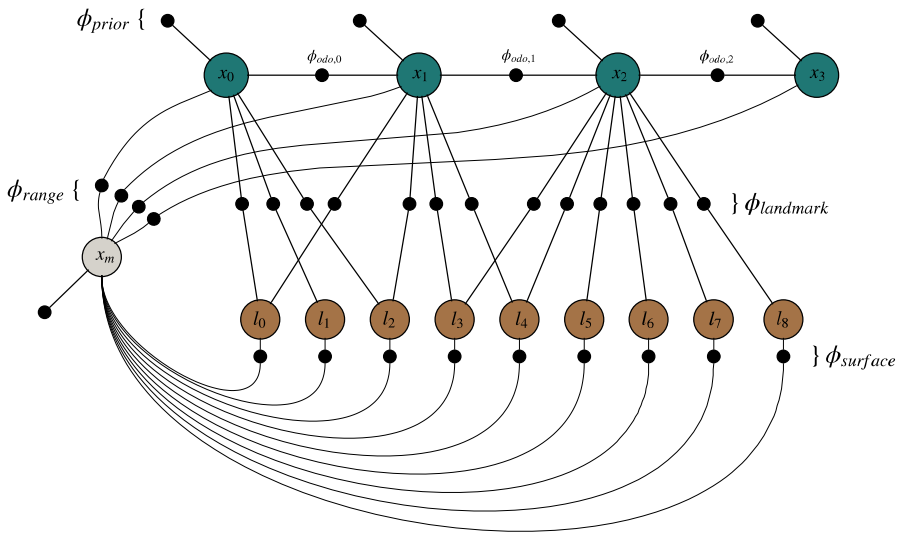


Figure 3.1: Factor graph with map prior.

face reconstruction, recently Geva et al. (2015) in which an UAV surveys a remote area. They used digital terrain models to regularise the position of 3D features observed from the camera mounted on the UAV.

Ozog et al. (2017) uses cad models of ship hulls as prior for mapping using BA using DVL and visual features. They show that their implementation is a special case of the Gaussian max-mixture models proposed by Olson and Agarwal (2013). See figure 3.1, the prior on the model position is created from the range measurements of the DVL, mapped to the model using generalised-icp by Segal et al. (2009).

Something similar could been experimented in our case, only we could use the result from the object detection as a prior for the detected object when inserted into the SLAM-map.

3.2 DSO

Direct sparse odometry by Engel et al. (n.d.) is the firs VO method to be both direct and sparse. Direct methods benefits of precise camera models including auto-exposure and gamma correction. Thus benefits more from the new cameras which are designed to be used in perception task.. DSO have performed very well above water. Direct methods, requires a good model and calibration of camera (Engel et al.; n.d.). Not only geometric properties, but photometric as well. This becomes complicated underwater, colour changes with depth and water conditions (Lu et al.; 2017). The paper states that good calibration is extremely important, our results does not reflect this, thus we would expect huge errors, and probably divergent solutions.

Table 3.2: Extensions of DSO

Year	Name	Citation	Extension	Source
2016	DSO	Engel et al.	Original	code
2018	LDSO	Gao et al.	Loop closure mechanisme	code
2018	Omnidirectional DSO	Matsuki et al.	Omnidirectional	
2018	VI-DSO	von Stumberg et al.	Incorporate IMU measurments	
2017	Stereo DSO	Wang, Schworer and Cremers	Stereo cameras	code
2018	Deep DSO	Yang et al.	Deep learning for depth prediction	

Matsuki et al. (2018) extended DSO to work with omnidirectional cameras. They state the importanse of having a large field of view (FOV) during VO/SLAM, as the large FOV will make it possible to track the same areas for longer, giving a more stable and robust result.

3.3 SVO

Semi-direct visual odometry by Forster et al. (2017, 2015). They try to combine the best of direct and indirect methods, by utilise all available information during tracking with directly minimising the photometric error of image pacthes in gradient rich areas. And the well defined BA for indirect methods by extracting features in keyframes.

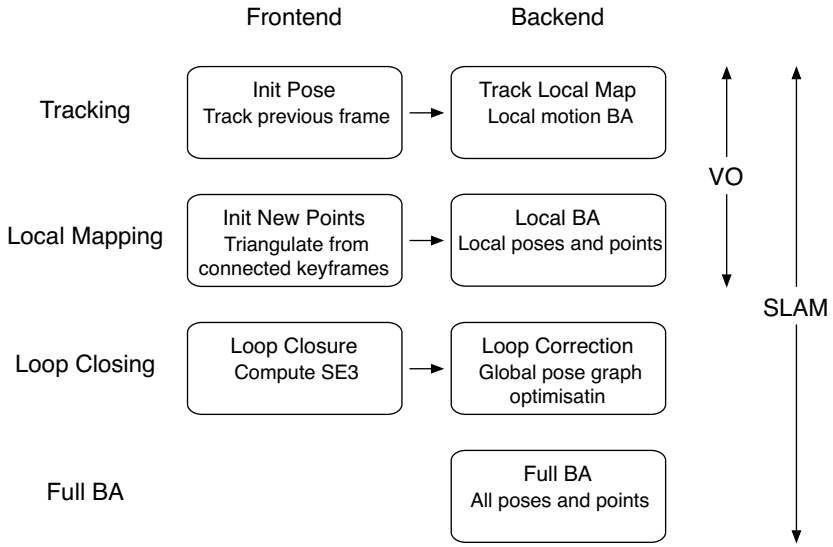


Figure 3.2: Distrobution between backend and frontend in ORB_SLAM2

3.4 ORB-SLAM

ORB-SLAM (Mur-Artal et al.; 2015; Mur-Artal and Tardos; 2016, 2017) is a sparse, indirect method, based on PTAM by Klein and Murray (2007). It uses ORB-features Rublee et al. (2011) for tracking, mapping and place recognition.

Figure 3.2 shows the different parts of ORB-SLAM, categorised by backend and frontend as well as how the SLAM problem is distributed in 4 threads to achieve real time operation. The first thread, tracking is running the fastest and has highest priority as it needs to run in real time. For each new image the tracking thread extracts ORB feature from the image and initialises the pose of the camera when the image was taken by minimising the reprojection error with points matched with the last frame only. Then using this as a initial pose it performs local motion bundle adjustment. The local mapping thread runs almost in real time. While the loop closing and full BA threads are running in the background updating the map once in a while when the

heavy optimisation have converged.

ORB-SLAM uses DBoW2 Galvez-López and Tardos (2012) for place recognition. DBoW2 is a bag of words method. Simply put it discretizes the Keyframes into a set of visual words by comparing the ORB-descriptors to a pre-trained bag of words. Then it stores the words of the Keyframe in a tree sorted by these visual words, such that it is very fast and easy to find Keyframes with a lot of the same visual words.

The backend uses the Levenberg-Marquardt implemented in g2o (Kummerle et al.; 2011) for factor graph optimisation.

The frontend uses max ratio test (Lowe; 1999) and cross-check as data-association enhancement.

Table 3.3: Extensions of ORB-SLAM

Year	Name	Citation	Extension	Source
2018	GVORB	Chen, Hu, Zhang, Shi and Li	Incorporate GNSS measurments	
2018	ORBSLAMM	Daoud et al.	Use multiple maps	code
2017	VIORB	Mur-Artal and Tardos	Incorporate IMU measurments	code
2017	PL-SLAM	Pumarola et al.	Include lines as features	
2016	ORB-SLAM2	Mur-Artal and Tardos	Extend to Stereo and RGB-D	code
2016	MultiCol-SLAM	Urban and Hinz	Multiple non-overlapping cameras	code
2015	ORB-SLAM	Mur-Artal et al.	Original	code

ORB-SLAM is a very popular method in the visual SLAM community where new methods often compare their results against ORB-SLAM. ORB-SLAM is also often used as a base, witch new papers extends or modify with their contributions. Some are listed in Table 3.3.

Urban and Hinz (n.d.) extended ORB-SLAM to use any rigidly coupled multi-camera systems, even if the cameras are not overlapping. They achieved this by modifying the key-frames to what they call Multi-Keyframes, where each Multi-Keyframes consists of the images from all the cameras. An interesting problem would be to extend this work to non-rigid multi-camera systems, where the cameras could move relative to each other, but with known movement. For instance where there are cameras which are placed both on the robot base and on a robot arm connected to the base.

To combat low textured scenes Pumarola et al. (2017) added lines as features to be used in the optimisation. They focused on computational efficiency by using LSD by von Gioi et al. (2012) for line detection and the relational graph strategy by Zhang and Koch (2013) for line matching.

Based on the theory of on-manifold preintegration by Forster et al. (2016), Mur-Artal and Tardos (2017) extends ORB-SLAM to, in a tightly manner, include IMU measurements. They add both velocities and IMU biases as nodes in the factor-graph optimisation problem discussed in chapter 2.

ORB-SLAM as well as most other of the current methods only operates with one map, one big global map. If the robot looses tracking and thus looses its location in the map, ORB-SLAM tries to relocate in the map by comparing new frames against old key-frames. If exploring new areas the robot often wont return to the same place it have been before, thus it will not be able to relocate its position, and be lost. Daoud et al. (2018) proposes a solution; instead of trying to relocate in the map after lost tracking, the robot simply initialises a new map, mark it as the active map and stores the old map. If it returns to a location in the old map, it then merges the two maps. Their method can also be used for multi-robot SLAM, where there are multiple robots exploring the same area, continuously sharing and merging their maps.

Lastly Chen, Hu, Zhang, Shi and Li (2018) incorporates GNSS measurements. Even though they add two additional threads to the system, they follow the same factor-graph representation and optimisation by adding GNSS measurements as priors to the Keyframes.

3.4.1 Discussion

Special designed cameras with exact mathematical models both for intrinsic as well as photometric combined with online estimation of the water state, like salinity, temperature, and pressure could give rise to direct underwater methods, which potentially could also track better than indirect methods in feature less areas. However until then, indirect methods have an advantage over direct methods because of the lack of precise camera models under water, which direct methods suffer more than indirect methods

(Engel et al.; n.d.). Thus we will focus on ORB-SLAM2 in this project, as ORB-SLAM2 is still considered state-of-the-art within indirect methods.

Chapter 4

Experiments

To improve and test our SLAM methods we need data sets or test facilities where we can evaluate and compare the results. For above water there exists several good public datasets like Burri et al. (2016) and Geiger et al. (2013). However for underwater use there exist, to the knowledge of the author, no public available datasets with ground truth. Codevilla et al. (2004) made public a dataset where they tested the robustness of features with regards to changing turbidity. They used a small thank with a image of the seabed in the bottom and a fixed camera pointed at the image, then they added milk to simulate turbidity. Ferrera et al. (n.d.) used the state of the art SFM method Colmap Schonberger and Frahm (2016) to estimate ground truth in real world images acquired with an ROV. Duarte et al. (2016) created a simulated dataset using the Underwater Simualtor[1] (UWSim) by Prats et al. (2012).

To generate ground truth for underwater one could use underwater cameras combined with reflective markers, but such systems are expensive and requires clear water. One could also use a fixed setup with rails but that is not mobile and is very restrictive for possible motions. We decided to try an acoustic positioning system. Unlike vision based systems it is capable of estimating position in turbid waters and it is mobile giving the possibility to do testing in different environments. It might

[1] `https://github.com/uji-ros-pkg/underwater_simulation`

not be as accurate as the two previously mentioned options, but when it comes to underwater SLAM we would argue that it is more important to evaluate robustness regarding different environments and turbidity than very precise accuracy.

4.1 Setup

We have performed our experiments in the pool at the Eelume lab. The pool is around 6 m long, 3.6 m wide and 1 m deep. See figure 4.1 and 4.3. An underwater GPS system from Waterlinked was placed in the pool.

4.1.1 Acoustic positioning

We have strategically placed the acoustic receivers for the underwater GPS system in the pool to get as big baselines as possible while avoiding metal objects and occlusions, see figure 4.1. We custom made wood structures to keep the receivers steady in the pool without damaging the pool surface, see figure 4.2.

d_1 [m]	d_2 [m]	d_3 [m]	d_4 [m]
0.43	0.42	0.38	0.39

Table 4.1: Measured depth for each receiver.

l_{12} [m]	l_{13} [m]	l_{14} [m]	l_{23} [m]	l_{24} [m]	l_{34} [m]
3.66	3.87	2.93	2.92	24.54	2.52

Table 4.2: Measured distance between the four receivers.

The coordinate system used for the Waterlinked underwater GPS system, *uwgps*, is defined with the xy-plane at the water surface, the z-axis perpendicular to the water surface pointing down, origo right above the position of receiver 1 and x-axis going along the water surface passing right above the position of receiver 2. See figure 4.3a. First we measure the depth of all the receivers, $d_i \quad i = 1, 2, 3, 4$, relative to the water

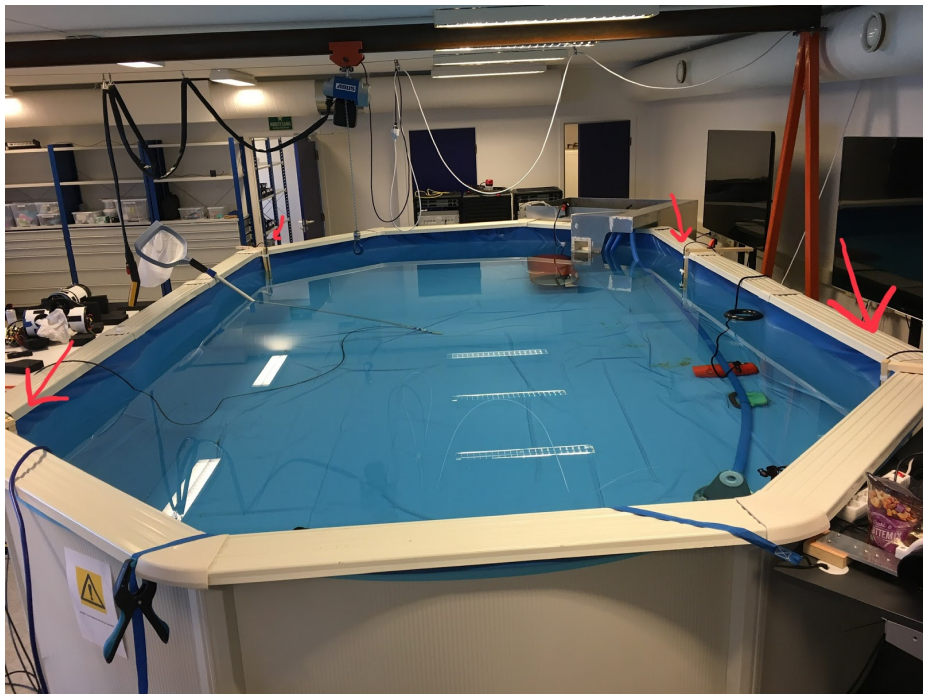


Figure 4.1: Pool at the Eelume lab used for experiments, with acoustic receivers marked with red arrows in the image.

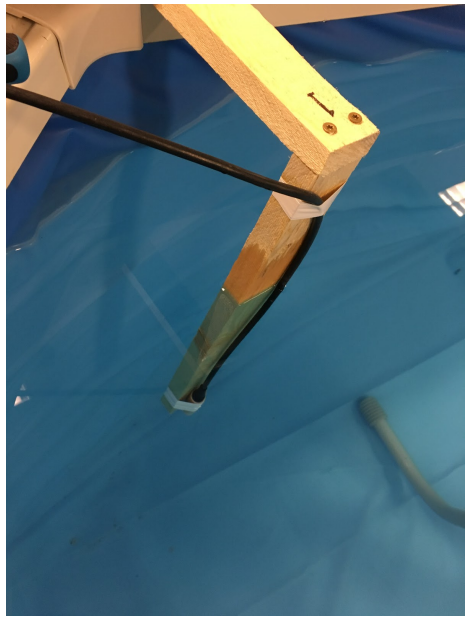


Figure 4.2: Placement of the acoustic receivers at around 0.4 m depth.

surface, see table 4.1. All the receivers are placed around 0.4 m depth, with the biggest difference being 5 cm. Then we measure the distance between all the receivers notated as l_{ij} for distance between receiver i and j, see table 4.2. These distances are between 2.5 m and 4.5 m, thus when calculating the xy-position of the receivers we assume they are laying at the same depth, i.e we do not take the difference in depth into account. Using this assumption and the defined *uwgps* coordinate system the position of receiver 1 and 2 are directly given, while 3 and 4 are found by simple triangulation:

$$p_1^{uwgps} = [0,\ 0,\ d_1]^\top \tag{4.1}$$

$$p_2^{uwgps} = [l_{12},\ 0,\ d_2]^\top \tag{4.2}$$

We use the law of cosines to find the angle between the line going through receiver 2 and 1, and the line going through receiver 1 and 3, see figure 4.3a.

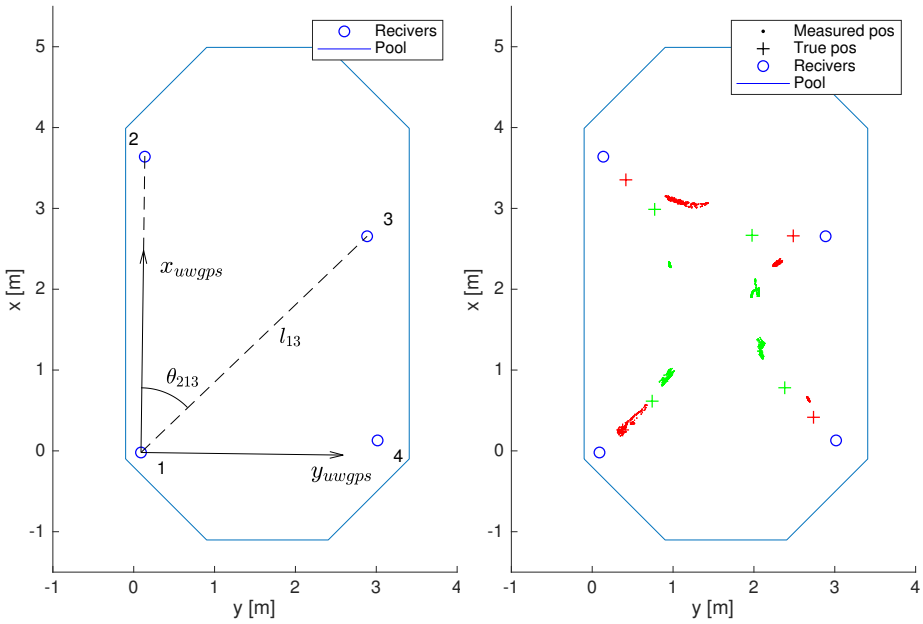
$$\theta_{213} = \arccos \frac{l_{12}^2 + l_{13}^2 - l_{23}^2}{2 l_{12} l_{13}} \tag{4.3}$$

This angle corresponds to the angle of the position vector of receiver 3 in the *uwgps* coordinate system. Thus the position is directly given as:

$$p_3^{uwgps} = [l_{13} \cos\theta_{213},\ l_{13} \sin\theta_{213},\ d_3]^\top \tag{4.4}$$

We calculate the position of receiver nr. 4 similarly, but by using the distance l_{14} instead of l_{13}. Finally we use the distance l_{34} as a quality check. Using our triangulated position of receiver 3 and 4, the distance l_{34} should be 252.4 cm while the measured distance was 252 cm, suggesting accurate measurements.

To evaluate the system with lack of ground truth we performed some static measurements. We placed the locator/transmitter at a fixed, known position, then we collected several measurements. See figure 4.3b. We performed two rounds of measurements, each round consisting of four measurements, one nearby each receiver. The first round, marked in red, we placed the transmitter close to the receiver. The second round, marked in green, we placed the locator closer to the middle of the



(a) Underwater gps coordinate system.

(b) Static measurements.

Figure 4.3: Underwater gps setup and experiment.

pool. As one can see from the figure we have a case of relative high precision, but low accuracy as there are in general a large bias, the mean error is as high as 0.47 m, but mostly low standard deviation, 0.1 m to 0.01 m. The bias tends towards the middle of the pool. One possible reason being that reflections from the walls causing perceived longer travel distances than reality. We did this experiment two times, the first time the receivers were as close to the walls of the pool as possible, the second time we moved the receivers approximately 30 cm towards the middle of the pool, but there were no significant difference in the performance. We have been in dialogue with Waterlinked, they inspected our setup and are working on software updates to increase performance for use in pools and tanks, therefor we decided to wait for the software update before performing more experiments.

4.1.2 Cameras

We have used two sets of stereo cameras, the first stereo camera consists of two FLIR (previosly Point Grey) Blackfly 1.3 MP GigE PoE cameras, further refereed to as just Blackfly. They have global shutter, 1288 x 964 pixel resolution at 30 FPS. The Blackfly cameras are placed inside custom made waterproof containers with a flat 3.6 mm Plexiglass in front, see figure 4.4a. We are using the ROS pointgrey camera driver[2] to feed the image streams into the ROS topics. We are using software synchronisation to trigger the cameras at the same time. However, what turned out to be a problem, the cameras are equipped with a magnifying lens.

The other set of cameras are two Teledyne Bowtech Divecam-720-AL, further refereed to as Divecam. They have 720x576 pixel resolution, rolling shutter, 25 FPS. They are made for underwater use with a 3.6 mm thick sapphire glass lens rated for 1000 m with a 91 ° diagonal field of view (FOV) in air and 65 ° in water. To create a stereo setup we mounted two of them to a piece of thick wood slightly pointing towards each other to increase the common field of view of the two cameras, see figure 4.4b. The Divecams are transmitting the images as analogue data, and are thus combined with a Z3 Technology FSDI-DCK-13 encoder. We use the ROS OpenCV

[2] https://github.com/ros-drivers/pointgrey_camera_driver

(a) Blackfly underwater stereo camera. (b) Teledyne Divecam stereo camera.

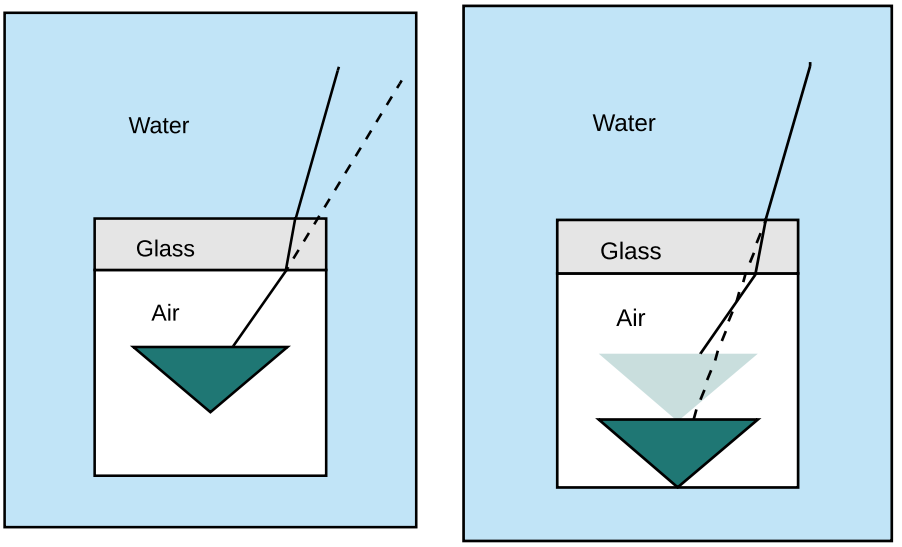
Figure 4.4: Underwater stereo cameras used for experiments. Both stereo cameras have a baseline of 20 cm.

video streamer[3] to publish the image streams to their ROS topics from the encoder.

4.2 Camera calibration

Underwater geometric camera models are a bit tricky due to the water-glass-air interface. The light rays changes direction when passing between materials with different refraction indexes. The normal approach have been to ignore this effect and absorb the errors into the standard camera parameters, particularly the radially lens distortion (Shortis; 2015). However Kunz and Singh (2008) argued that this is a significant modelling error, particularly if the object of interests are at varying distances from the camera. A more general geometric correction solution is developed for plane port housings in: Jordt-Sedlazeck and Koch (2012) where they employ a two step approach of first calibrating the camera on land, then calibrate the housing underwater. Łuczyński et al. (2017) presents Pinax Model as it combines aspects of a virtual pinhole model with the projection function from the axial camera model. See figure 4.5b, the stapled line represents the refraction corrected light-ray. It allows pre-computation of a lookup-table for very fast refraction correction of the flat-pane with high accuracy. The model takes the refraction indices of water into account, especially with respect to salinity, and it is therefore sufficient to calibrate the underwater camera only once in air. It is demonstrated by real world experiments with several underwater cameras

[3] `http://wiki.ros.org/video_stream_opencv`



(a) Pinhole model. The stapled line represent the estimated projection using a Pinhole model, the full line represents the real projection.

(b) Pinax model. The stapled line represents the remapped projection into an virtual pinhole model.

Figure 4.5: Water-glass-air interface refraction. The light-rays changes direction when they passes between the different materials.

in different salt and sweet water conditions that the proposed process outperforms standard methods. Among others, it is shown how the presented method leads to accurate results with single in-air calibration and even with just estimated salinity values.

We have gathered 4 calibration sets, two for each stereo camera, one above water and one under water. Then we have for both cameras tested two camera models:

1. The Pinhole model with radial distortion.
2. The Pinax model by Łuczyński et al. (2017).

At last we compared the reprojection error for the two models on the underwater

datasets.

4.2.1 Calibration data-set

For the Blackfly stereo setup we took around 100 images with 2 seconds interval while we kept the camera fixed and moved the calibration board around, both in air and underwater. While for the Divcam stereo setup we used the default calibration method in ROS which is based on opencv. While one move the calibration board around, the ROS calibration algorithm will automatically and in real time save the images it find good enough. Thus these data sets contains less images, 30 and 40 images for underwater and in air respectively, but each image contains more unique information compared to the Blackfly calibration set.

Unless the used camera model is perfect, it will often perform best when the objects of interest typically are in the same distance from the camera as the calibration pattern was during the calibration. The Blackfly cameras are equipped with magnifying lenses thus we needed to keep the calibration board at least 2.5 m distance in order to get the entire board in on image, even further to get the entire board in both the left and right images from the stereo camera. From figure 4.7 we also see that the spread in distance from the camera is larger for the Blackfly camera than the Divecam. This is largely because the Blackfly has higher resolution images, and less field of view, thus it is able to accurately detect the calibration pattern all the way across the pool. The Divecam has a wider field of view and less resolution thus it only manged to detect the calibration pattern at maximum 2.5 meters distance.

We used the ROS camera calibration node when gathering the calibration set for the Divecam, where every image of the calibration board where checked live as we recorded, and if it could not detect the calibration pattern it did not save the image. We suspect that this calibration pattern detector, which ran in real time while we recorded the images, performed worse than the ones used later (Matlab and Kalibr) when we did a deeper analysis, but since it only saved the images when it could detect the pattern, we can not verify this hypothesis before we can record new images. In other words we discarded a lot of the raw data from the experiments too soon and as a result the

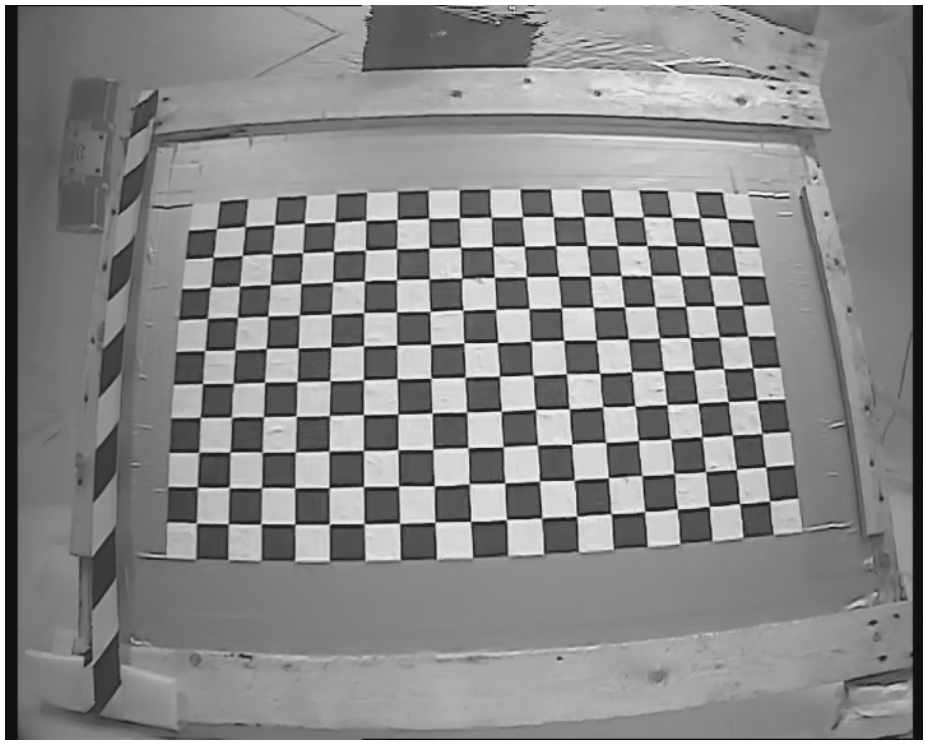
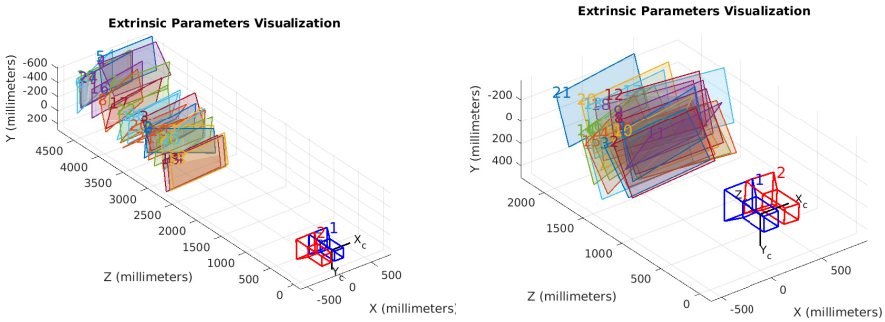


Figure 4.6: Calibration board used. A machine painted 18x10 chessboard pattern, with 5x5 cm squares, on a glass plate. The paint started to fall off so we kept it together by using black and white tape, we made sure to only tape within the squares to not effect the detected pattern.



(a) Blackfly underwater stereo camera.

(b) Teledyne divecam stereo camera.

Figure 4.7: Extrinsic visualisation of the underwater calibration data sets. Note that for the Blackfly calibration set the distance between the calibration board and the camera varies almost 2.5 m, while for the Divecam it only vary around 1 m. Also note that because of the magnifying lens mounted on the Blackfly cameras it can not detect the calibration board at distances closer than 2.5 meter.

spread of detected calibration patterns in the Divecam calibration set probably could have been better, and should be improved in future work. But we learned an important lesson, that one should be extremely careful in disregarding and removing raw data from experiments.

4.2.2 Pinhole

The Pinhole camera model is the oldest and most used camera model, with only 4 parameters it is very simple, but effective. We have used it in combination with a radial distortion model with two parameters. For underwater use it is fundamentally wrong as it do not take into account the refraction caused by the water-to-glass and air-to-glass interface, however as demonstrated in Łuczyński et al. (2017) it will still work reasonably well if the object of interest stays at a fixed distance from the camera and the pinhole model is calibrated for this exact distance (by keeping the calibration board at this distance during calibration).

4.2.3 Pinax

As an alternative to the Pinhole model we have tested the Pinax model by Łuczyński et al. (2017) for our underwater camera model. The Pinax model is a geometric camera model for underwater camera systems where the camera is shielded from the water by a flat-pane interface, e.g. a glass window. It is a combination of the pinhole camera model and the axial camera model. Unlike most methods used for underwater geometric camera calibration it only needs one in-air calibration. Then it is able to calculate the underwater projection matrix and distortion effects based on four physical values which needs to be known:

- d_0 - Distance between the center of projection of the camera and the glass window.
- d_1 - Thickness of the glass.
- n_g - Refraction index of the glass.
- n_w - Refraction index of the water.

Table 4.3: Parameters used in the Pinax model for the Divecam stereo setup. and Austin and Halikas (1976). The same parameters where used for both left and right cameras in each stereo setup.

	w [pix]	h [pix]	d_0 [mm]	d_1 [mm]	n_g	n_w
Divecam	720	676	0.5	3.6	1.768	1.334
Point Grey	1288	964	8	19	1.492	1.334

Table 4.3 shows the parameters used for the Pinax model, the glass used in the Divecam is made of Sapphire glass while the one used in the Blackfly is Plexiglass and the refraction indexes where found from `https://refractiveindex.info`. The water refraction index is taken from Austin and Halikas (1976), where we estimated salinity to be 0.5 ppm and the temperature to be 20 °C. d_0 was estimated using a Matlab-script provided by Łuczyński et al..

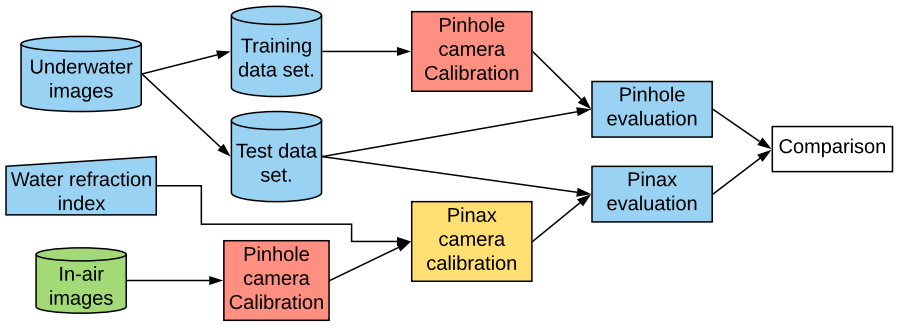


Figure 4.8: Camera calibration data flow. We want to compare the Pinax model vs in-situ calibrated pinhole model. Since the pinhole model is estimated directly from the underwater calibration set we split the set in two and use one for training and one for testing. For the test set we only removed images which qualified as outliers for both the Pinax and the Pinhole model.

4.2.4 Comparison

We estimated the pinhole model in-situ, that is directly from the underwater calibration images, which are the same images we used to compare the Pinhole model and the Pinax model. To prevent the Pinhole model from over-fitting and thus create an unfair comparison with the Pinax model we split the data set in two equal parts keeping somewhat the same calibration board poses in each part. Then we used one of the data sets for training and one for testing. See figure 4.8. Since the pinhole model with radial distortion only consists of 6 parameters the chance of over fitting is minimal, but the calibration software typically regards the images with the worst re-projection error as outliers and removes them, thus drastically increase the chance of over-fitting. For the test set we thus only removed the images where both the Pinax and the Pinhole model scored particularly bad, which was around 20-30 % of the images and mostly due to motion blur and poor lightning.

Table 4.4 shows the results from calibration of the underwater stereo setup equipped with Blackfly cameras. Firs row shows the mean and standard deviation in pixels from the in-air calibration of a Pinhole model. The second row shows the results of the

	Mean [pix]	Standard deviation [pix]
In-air	0.249	0.114
Pinax	0.212	0.084
In-Situ Pinhole	0.401	0.159

Table 4.4: Reprojection errors from the calibration of the Blackfly stereo camera setup.

	Mean [pix]	Standard deviation [pix]
In-air	0.361	0.031
Pinax	1.513	0.391
In-Situ Pinhole	0.122	0.027

Table 4.5: Reprojection errors from the calibration of the Divecam stereo camera setup.

Pinax model on the underwater evaluation set, note that these results are based on the results from the in-air calibration as they are used to calculate the Pinax model. The last row displays the results from the Pinhole model calibrated on underwater images. It is a bit surprising that the In-air Pinhole reprojection errors are higher than from the underwater Pinax model, it could be explained by the fact that the FOV is quite larger in air than underwater, but the difference is less than a half standard deviation, and not very relevant. Further we note that the Pinax model score almost twice as good as the Pinhole model under water.

Table 4.3 shows the results from calibrating the stereo setup consisting of two Divecam cameras in their original enclosure. Here the Pinax model is not as promising, it is possible the assumption of a flat lens was wrong.

4.3 SLAM

The Blackfly camera turned out to have too much zoom to be able to be used for visual SLAM in the pool, thus we only used the Divecam stereo rig for testing visual SLAM in the pool. Since we do not yet have a reliable acoustic positioning system to use as ground truth, we recorded two data set where we started and ended in the exact same

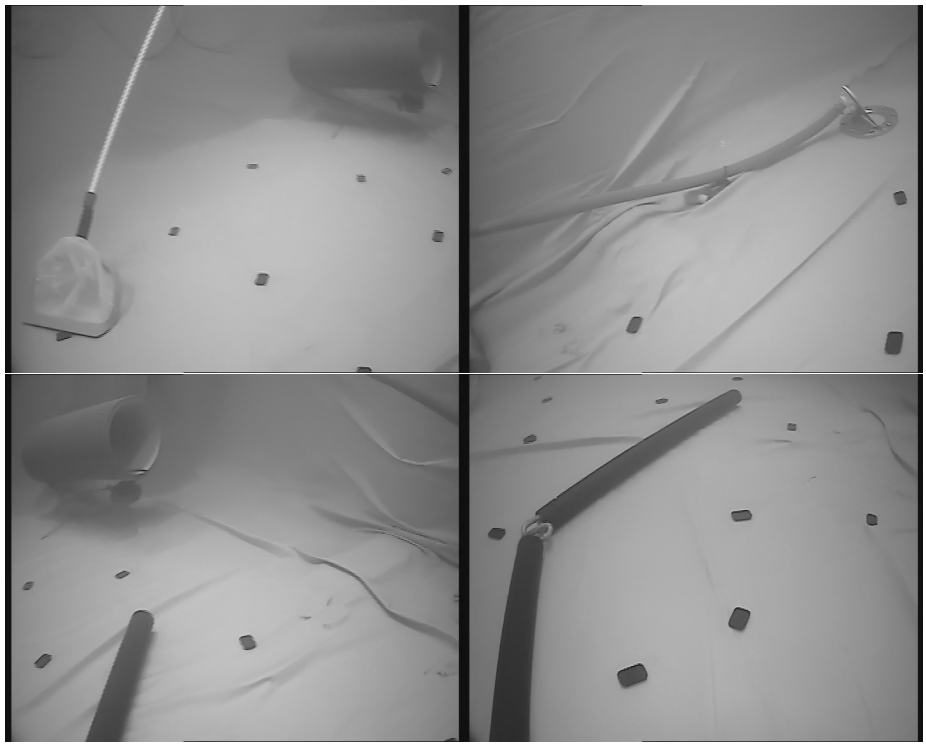


Figure 4.9: Images from the datasets, we used small black metal pieces to create some texture in the enivironment.

position. We achieved this by leaning the camera rig towards a ladder placed in the pool at the beginning and end of the each run. Using these data sets we can:

- Do qualitative evaluation of the map.
- Verify the loop closure mechanism.
- Quantify drift by turning off loop closure and measure the difference between start and end pose.

First we tested SVO 2, but it was not able to complete any of the datasets, most likely due to the fact that the cameras does not have global shutter, and we used Pinhole

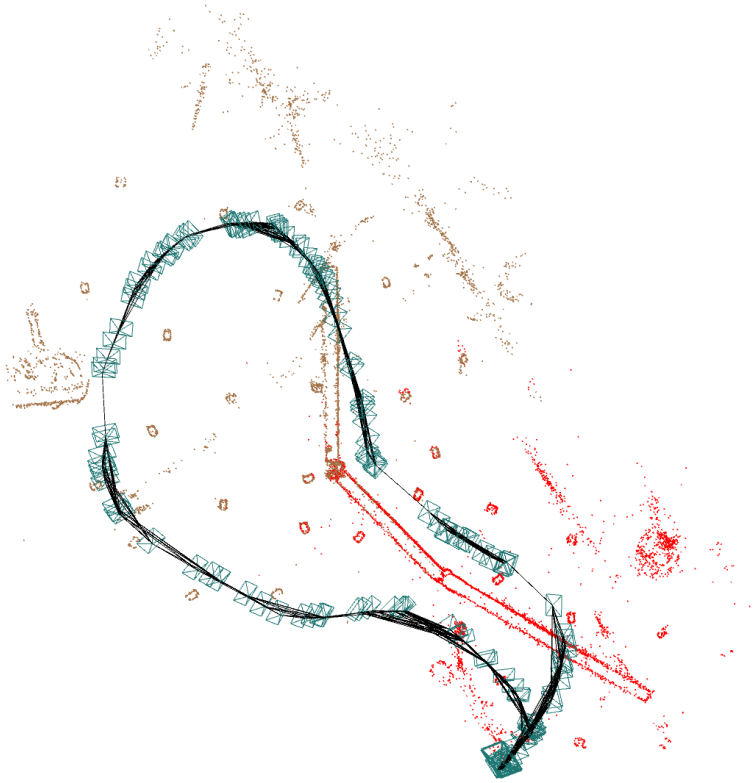
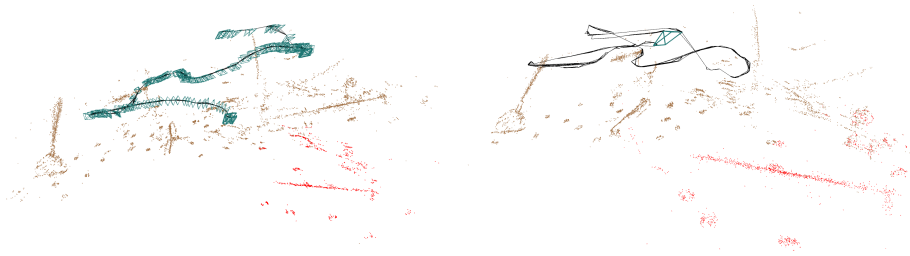


Figure 4.10: Resulting map and trajectory from running ORB-SLAM2 on dataset 1. The brown and red points represents the map, while the green frames represents the Keyframes and the black lines represent the connectivity graph.



(a) Before loop closure. (b) After loop closure.

Figure 4.11: Loop closure in dataset 2.

model under water, resulting in a relative inaccurate camera model. An accurate camera model is very critical for direct methods, as shown in Engel et al. (n.d.).

Second we tested ORB-SLAM2, which is an indirect method, and thus more robust to rolling shutter and inaccurate camera models. The results follow: The numerical values was found by taking the median out of 5 runs, the parameters for ORB-SLAM2, are as following: we used the Pinhole camera model, (We tried the Pinax, but was only able to complete the second data set, and got significantly worse results than the Pinhole). Further we changed the feature thresholds to: 1000 features per image, initial FAST feature threshold set to 20, and minimum set to 2.

Sequence one is approximately 9 meter long, this estimate is found by using the resulting trajectory after loop closure, the camera is for the most of the time in the surface pointing down towards the bottom and the movement is rather slow. Before loop closure the distance between the end and start position of the camera is 0.25 meters, resulting in approximately 2.8 % error which is quite impressive.

Sequence two is approximately 11 meter long, the camera is moving around in the water changing between pointing downwards, forward and sideways, the movement is faster than in the first dataset. Before loop closure the distance between the end and start position of the camera is 0.70 meters, resulting in approximately 6.3 % error.

For each dataset ORB-SLAM2 managed to detect and close the loop, see figure 4.11, after returning to the same position 7/10 times. Even though the environment was

very similar around the pool, this should have been higher since the camera returned to the exact same position every time.

Chapter 5

Conclusions and future work

In this project we have

- Set up and tested an underwater acoustic positioning system, including programming a ROS driver to publish the resulting measurements to a ROS topic.
- Set up two different underwater stereo camera systems, both with different ROS drivers, one of them with software synchronisation.
- Calibrate both the camera systems using two camera models, the Pinhole and the Pinax model.
- Recorded two datasets and tested two SLAM methods. One which shows quite promising results.

In the final result we used the Pinhole camera model, which is not accurate for underwater use and a stereo camera which was not properly synchronised, but still ORB-SLAM 2 performed good. This showcase the robustness of indirect methods towards modelling errors. However the movement of the camera was quite slow, and we worked in an ideal environment with clear water and artificially structures with sharp corners. In the future we will improve our camera setup by ordering new wide-angle lenses for the Blackfly cameras. Further we will continue working on

ORB-SLAM2, extending it to include IMU measurements and depth measurements, hopefully making it more robust. We have already gotten hold of an IMU and pressure sensor and are working on creating a waterproof enclosure. Then the next step will be to combine the SLAM method with the planning and control system as well as the object detection part of the bigger project, to create a more complete autonomous system, capable of performing simple tasks such as hovering and follow a pipeline. We also want to get the acoustic positioning system to work properly. We can either use it as ground truth or as an extra sensor to be used for a even more robust SLAM system.

References

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M. and Szeliski, R. (2011). Building rome in a day, *Communications of the ACM* **54**(10): 105.

Austin, R. W. and Halikas, G. (1976). The index of refraction of seawater.

Bryson, M., Johnson-Roberson, M., Pizarro, O. and Williams, S. B. (2013). Colour-consistent structure-from-motion models using underwater imagery, *Robotics: Science and Systems VIII* p. 33.

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

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, *IEEE Transactions on Robotics* **32**(6): 1309–1332.

Chen, C., Zhu, H., Li, M. and You, S. (2018). A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives, *Robotics* **7**(3): 45.

Chen, X., Hu, W., Zhang, L., Shi, Z. and Li, M. (2018). Integration of low-cost gnss and monocular cameras for simultaneous localization and mapping, *Sensors* **18**(7): 2193. Incorporate GNSS measurments.

Codevilla, F., Gaya, J. D. O. and Botelho, S. (2004). Achieving turbidity robustness on underwater images local feature detection, *International journal of computer vision* **60**(2): 91–110.

Corke, P., Paul, R., Churchill, W. and Newman, P. (2013). Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation, *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE.

Daoud, H. A., Sabri, A. Q. M., Loo, C. K. and Mansoor, A. M. (2018). SLAMM: Visual monocular SLAM with continuous mapping using multiple maps, *PLOS ONE* **13**(4): e0195878. Use multiple maps.

Delmerico, J. and Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots.

Duarte, A. C., Zaffari, G. B., da Rosa, R. T. S., Longaray, L. M., Drews, P. and Botelho, S. S. C. (2016). Towards comparison of underwater SLAM methods: An open dataset collection, *OCEANS 2016 MTS/IEEE Monterey*, IEEE.

Engel, J., Koltun, V. and Cremers, D. (n.d.). Direct sparse odometry.

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

Forster, C., Carlone, L., Dellaert, F. and Scaramuzza, D. (2015). IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation, *Robotics: Science and Systems XI*, Robotics: Science and Systems Foundation.

Forster, C., Carlone, L., Dellaert, F. and Scaramuzza, D. (2016). On-manifold preintegration for real-time visual–inertial odometry, *IEEE Transactions on Robotics* **33**(1): 1–21.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M. and Scaramuzza, D. (2017). SVO: Semidirect visual odometry for monocular and multicamera systems, *IEEE Transactions on Robotics* **33**(2): 249–265.

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

Gao, X., Wang, R., Demmel, N. and Cremers, D. (n.d.). Ldso: Direct sparse odometry with loop closure.

Geiger, A., Lenz, P., Stiller, C. and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset, *The International Journal of Robotics Research* **32**(11): 1231–1237.

Geva, A., Briskin, G., Rivlin, E. and Rotstein, H. (2015). Estimating camera pose using bundle adjustment and digital terrain model constraints, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE.

Gracias, N., Negahdaripour, S., Neumann, L., Prados, R. and Garcia, R. (2008). A motion compensated filtering approach to remove sunlight flicker in shallow water images, *OCEANS 2008*, IEEE.

Hidalgo, F. and Braunl, T. (2015). Review of underwater SLAM techniques, *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE.

Huang, S. and Dissanayake, G. (2016). A critique of current developments in simultaneous localization and mapping, *International Journal of Advanced Robotic Systems* **13**(5): 172988141666948.

Jordt-Sedlazeck, A. and Koch, R. (2012). Refractive calibration of underwater cameras, *Computer Vision – ECCV 2012*, Springer Berlin Heidelberg, pp. 846–859.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces, *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE.

Kschischang, F., Frey, B. and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm, *IEEE Transactions on Information Theory* **47**(2): 498–519.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W. (2011). Gsup2/supo: A general framework for graph optimization, *2011 IEEE International Conference on Robotics and Automation*, IEEE.

Kunz, C. and Singh, H. (2008). Hemispherical refraction and camera calibration in underwater vision, *OCEANS 2008*, IEEE.

Lowe, D. (1999). Object recognition from local scale-invariant features, *Proceedings of the Seventh IEEE International Conference on Computer Vision*, IEEE.

Lowry, S., Sunderhauf, 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.

Lu, H., Li, Y., Zhang, Y., Chen, M., Serikawa, S. and Kim, H. (2017). Underwater optical image processing: a comprehensive review, *Mobile Networks and Applications* **22**(6): 1204–1211.

Łuczyński, T., Pfingsthorn, M. and Birk, A. (2017). The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings, *Ocean Engineering* **133**: 9–22.

Massot-Campos, M. and Oliver-Codina, G. (2015). Optical sensors and methods for underwater 3d reconstruction, *Sensors* **15**(12): 31525–31557.

Matsuki, H., von Stumberg, L., Usenko, V., Stuckler, J. and Cremers, D. (2018). Omnidirectional DSO: Direct sparse odometry with fisheye cameras, *IEEE Robotics and Automation Letters* **3**(4): 3693–3700.

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. Original.

Mur-Artal, R. and Tardos, J. D. (2016). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras, *IEEE Transactions on Robotics* **33**(5): 1255–1262. Extend to Stereo and RGB-D.

Mur-Artal, R. and Tardos, J. D. (2017). Visual-inertial monocular SLAM with map reuse, *IEEE Robotics and Automation Letters* **2**(2): 796–803. Incorporate IMU measurments.

Olson, E. and Agarwal, P. (2013). Inference on networks of mixtures for robust robot mapping, *The International Journal of Robotics Research* **32**(7): 826–840.

Oth, L., Furgale, P., Kneip, L. and Siegwart, R. (2013). Rolling shutter camera calibration, *2013 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE.

Ozog, P., Johnson-Roberson, M. and Eustice, R. M. (2017). Mapping underwater ship hulls using a model-assisted bundle adjustment framework, *Robotics and Autonomous Systems* **87**: 329–347.

Paull, L., Saeedi, S., Seto, M. and Li, H. (2014). AUV navigation and localization: A review, *IEEE Journal of Oceanic Engineering* **39**(1): 131–149.

Prats, M., Perez, J., Fernandez, J. J. and Sanz, P. J. (2012). An open source tool for simulation and supervision of underwater intervention missions, *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE.

Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A. and Moreno-Noguer, F. (2017). PL-SLAM: Real-time monocular visual SLAM with points and lines, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE. Include lines as features.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF, *2011 International Conference on Computer Vision*, IEEE.

Saputra, M. R. U., Markham, A. and Trigoni, N. (2018). Visual SLAM and structure from motion in dynamic environments, *ACM Computing Surveys* **51**(2): 1–36.

Schonberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE.

Segal, A., Haehnel, D. and Thrun, S. (2009). Generalized-icp, *Robotics: science and systems* **2**(4): 435.

Shortis, M. (2015). Calibration techniques for accurate measurements by underwater camera systems, *Sensors* **15**(12): 30810–30826.

Urban, S. and Hinz, S. (n.d.). Multicol-slam - a modular real-time multi-camera slam system. Multiple non-overlapping cameras.
URL: *https://arxiv.org/pdf/1610.07336.pdf*

von Gioi, R. G., Jakubowicz, J., Morel, J.-M. and Randall, G. (2012). LSD: a line segment detector, *Image Processing On Line* **2**: 35–55.

von Stumberg, L., Usenko, V. and Cremers, D. (n.d.). Direct sparse visual-inertial odometry using dynamic marginalization.

Wang, R., Schworer, M. and Cremers, D. (2017). Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras, *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE.

Wang, Y., Zhang, W. and An, P. (2017). A survey of simultaneous localization and mapping on unstructured lunar complex environment, Author(s).

Yang, N., Wang, R., Stückler, J. and Cremers, D. (n.d.). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry.
URL: *http://arxiv.org/pdf/1807.02570v2*

Zhang, L. and Koch, R. (2013). An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency, *Journal of Visual Communication and Image Representation* **24**(7): 794–805.