Monocular Visual-Inertial-Pressure Fusion for Underwater Localization and 3D Mapping.

Maxime Ferrera

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITE DE MONTPELLIER

En Systèmes Automatiques et Micro-Électroniques

École doctorale : Information, Structures, Systèmes

Unité de recherche LIRMM / ONERA (DTIS - IVA)

Monocular Visual-Inertial-Pressure Fusion for Underwater Localization and 3D Mapping.

Présentée par Maxime Ferrera
le 12 décembre 2019

Sous la direction de Vincent Creuze,
Julien Moras et Pauline Trouvé-Peloux

Devant le jury composé de

Khatib Oussama	Professeur	Stanford University	Président du jury
Garcia Rafael	Professeur	Universitat de Girona	Rapporteur
Jaulin Luc	Professeur	ENSTA Bretagne	Rapporteur
Filliat David	Professeur	ENSTA Paris	Examinateur
L'Hour Michel	Directeur du DRASSM	Ministère de la Culture	Examinateur
Creuze Vincent	Maître de Conférence	Université de Montpellier	Directeur de thèse
Moras Julien	Ingénieur de Recherche	ONERA	Encadrant
Trouvé-Peloux Pauline	Ingénieure de Recherche	ONERA	Encadrante

UNIVERSITÉ
DE MONTPELLIER

Remerciement

Je souhaite tout d'abord remercier les professeurs Luc Jaulin et Rafael Garcia d'avoir accepté de rapporter ce manuscrit de thèse. Je remercie également les professeurs Oussama Khatib et David Filliat, ainsi que Michel L'Hour, pour avoir accepté de faire partie de mon jury de thèse en tant qu'examinateurs.

Je remercie Vincent Creuze, mon directeur de thèse, ainsi que Julien Moras et Pauline Trouvé-Peloux, mes encadrants à l'ONERA, pour m'avoir accompagné tout au long des ces trois années de thèse. Merci également pour toutes les opportunités d'expérimenations, qui ont été un facteur plus que motivant pendant cette thèse.

Un grand merci à toute l'équipe de l'ONERA, DTIM / IVA, dans laquelle j'ai eu la chance de pouvoir travailler, tout d'abord à l'occasion d'un stage fin d'étude, puis pendant ces trois années de thèse. Merci à Julien et Aurélien pour m'avoir encadré pour la durée d'un stage qui s'est finalement conclu sur un début de thèse. Merci à tous les membres du DTIM de cette époque: Guy, Fred, Fabrice J., Elise, Aurélien, Benjamin, Alex B., Adrien, Phillipe, Martial, Alex E., Anthelme, Stéphane, Bertrand, Jean, Christian, Jérome, Eric, Olivier, Patrick, Alain, Fabrice, Gilles. Merci également à Françoise pour les pauses cafés croustillantes de cette époque. Je remercie plus particulièrement Alex et Martial pour les discussions sur le SLAM et Alex B. pour les collaborations en Deep Learning. Merci aussi Alex E., Martial et Julien pour n'avoir jamais hésité à se plonger dans les méandres du C++ et d'UNIX lors de problèmes techniques on ne peut moins appétissant !

Merci aussi à tous les doctorants, stagiaires, apprentis de cette époque : Sémy, Maxime B., Calum, David, Hicham, Nicolas, Maxime D., Isabelle, Guillaume, Joris, Orianne et Marcela ! Merci à la tireuse pour nous avoir accueillis plus que régulièrement à cette époque là !

La continuité en thèse était loin d'être une évidence pour moi et vous êtes tous un peu responsable de mon choix de part votre sympathie et les très bonnes conditions de travail (intellectuement et socialement parlant)!

Je remercie également les doctorants qui sont arrivés entre temps : Guillaume VR., Pierre, Alexis, Rémy, Gaston, Soufiane, Rodriguo, Guillaume, Rodolphe, Lauranne et Javiera ! Bon courage à tous pour les années de thèse qu'ils vous restent !

Je souhaite également remercier tous les copains de MTS, les gars d'Orléans et la Team d'Ivry! Je remercie aussi ma famille, Alex et Anaïs, ainsi que mes parents pour m'avoir toujours soutenu et encouragé. Je suis de plus très reconnaissant à mes parents pour avoir eu la clairvoyance de m'inciter à me tourner vers des études scientifiques plutôt que de commerce lors d'un moment de perdition en terminale.

Enfin merci à Yolaine pour m'avoir supporter (dans les deux sens !) tout au long de cette thèse.

Et parce qu'on n'est plus à un merci près: MERCI A TOUS!

Abstract

This thesis addresses the problem of real-time 3D localization and mapping in underwater environments. In the underwater archaeology field, Remotely Operated Vehicles (ROVs) are used to conduct deep-sea surveys and excavations. Providing both accurate localization and mapping information in real-time is crucial for manual or automated operation of the robots. While many localization solutions already exist for underwater robots, most of them rely on very accurate sensors, such as Doppler velocity logs or fiber optic gyroscopes, which are very expensive and may be too bulky for small ROVs. Acoustic positioning systems are also commonly used for underwater positioning, but they provide low frequency measurements, with limited accuracy. In this thesis, we study the use of low-cost sensors for accurate underwater localization. Our study investigates the use of a monocular camera, a pressure sensor and a low-cost MEMS-IMU as the only means of performing localization and mapping in the context of underwater archaeology. We have conducted an evaluation of different features tracking methods on images affected by typical disturbances met in an underwater context. From the results obtained with this evaluation, we have developed a monocular Visual SLAM (Simultaneous Localization and Mapping) method, robust to the specific disturbances of underwater environments. Then, we propose an extension of this method to tightly integrate the measurements of a pressure sensor and an IMU in the SLAM algorithm. The final method provides a very accurate localization and runs in real-time. In addition, an online dense 3D reconstruction module, compliant with a monocular setup, is also proposed. Two lightweight and compact prototypes of this system have been designed and used to record datasets that have been publicly released. Furthermore, these prototypes have been successfully used to test and validate the proposed localization and mapping algorithms in real-case scenarios.

Résumé

Cette thèse aborde le problème de la localisation et cartographie 3D sous-marine en temps-réel. Dans le domaine de l'archéologie sous-marine, des véhicules téléopérés (ROV – Remotely Operated Vehicle) sont utilisés pour étudier les sites. La localisation et la cartographie précises en temps-réel sont des informations essentielles pour le pilotage manuel ou automatique de ces engins. Bien que plusieurs solutions de localisation existent, la plupart d'entre elles reposent sur l'utilisation de capteurs tels que les lochs Doppler (DVL – Doppler Velocity Log) ou les centrales inertielles à gyroscopes à fibre optique, qui sont très coûteux et peuvent être trop volumineux ou trop lourds pour les ROVs les plus petits. Les systèmes de positionnement acoustique sont également fréquemment utilisés en complément des systèmes précédents, mais leur fréquence d'échantillonnage et leur précision sont limitées. Dans cette thèse, nous étudions l'utilisation de capteurs à faible coût pour la localisation sous-marine de précision. Notre étude porte sur l'utilisation d'une caméra monoculaire, d'un capteur de pression et d'une centrale inertielle MEMS (Micro ElectroMechanical System) à faible coût comme seul moyen de localisation et de cartographie en contexte archéologique sous-marin. Nous avons mené une évaluation de différentes méthodes de suivi de point d'intérêts sur des images affectées par des perturbations typiques rencontrées dans un contexte sous-marin. À partir des résultats obtenus nous avons développé une méthode monoculaire de SLAM (Simultaneous Localization and Mapping) robuste aux perturbations spécifiques de l'environnement sous-marin. Ensuite, nous proposons une extension de cette méthode pour intégrer étroitement les mesures du capteur de pression et de la centrale inertielle dans l'algorithme de SLAM. La méthode finale fournit une localisation très précise et s'exécute en temps-réel. En outre, un module de reconstruction 3D dense, en ligne, compatible avec une configuration monoculaire, est également proposé. Deux prototypes compacts et légers de ce système ont été conçus et utilisés pour enregistrer des jeux de données qui ont été publiés. En outre, ces prototypes ont été utilisés avec succès pour tester et valider en conditions réelles les algorithmes de localisation et de cartographie proposés.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Context

What lies beneath the oceans is mostly unknown to humankind. This is mainly due to the difficult access of the underwater world to humans. However, Mars and the Moon are hostile environments too. The impossibility of large scale visualization, unlike satellite imaging for instance, makes the study of the seabed much more difficult and much longer. Indeed, the most efficient way to discover *new worlds* is to visualize them. The development of cameras has allowed humans to see beyond what they could naturally see and act as an artificial extension of humans' eyes. By equipping satellites with high-end optical systems or sending high-end telescopes into space, images of the whole Earth surface and of extra-terrestrial worlds have been acquired, making it possible for humans to visualize and better understand them. However, the physical principles of cameras (considering the ones that measure visible wavelengths) relies on the measurement of the reflexion of light on object surfaces. The problem with oceans's floor imaging is that oceans's water absorbs light very fast. After a few hundreds of meter below the surface, natural light has completely vanished. Therefore, it is not possible to acquire images of the oceans' floors at great depths from the surface. A solution is to use sonars, whose acoustic waves well propagate into water (even in turbid water). However, the images produced by such systems do not provide as much information as cameras and are not sufficient for accurate analysis.

The main target application of this thesis is underwater archeology. In this field, very accurate information about what lies on the oceans' floor are required. Sonar systems are useful for the detection of potential archaeological sites (mostly shipwrecks), but archaeologists need optical images of these wrecks to understand and analyze them. In shallow waters (usually at less than 50 meters), divers can perform the survey and excavation in person. However, for wrecks located beyond the limits of human diving, archaeologists have to use underwater robots. These machines are used as remote eyes and arms to study the wrecks and sample some artifacts for further analysis.

In underwater robotics, we find two classes of robots: Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs). AUVs are fully autonomous robots. Therefore, they have to embed their own batteries and a set of sensors allowing them to navigate autonomously. They are useful for long-range navigation, but remain difficult to use in areas requiring cautious navigation. On the other hand, ROVs are tethered robots that can be tele-operated. They do not have any battery issue as they usually get their energy through the tether. ROVs are most of the time deployed and piloted from ships. Their versatility allows a wide range of operations, even in complex environments.

ROVs are the most used kind of underwater robots because of their teleoperation capability. They can be directly embedded on ships and then deployed on the areas of interest. They are then tether to the ship and piloted from on-board in order to perform the required tasks. They are very useful for operations near the seabed, that requires precise and cautious navigation. ROVs are used in many fields (Oil&Gas Industry, telecoms, mine warfare archaeology, biology) either for observation or manipulation purposes. Different classes of ROVs exist, with different targeted applications:

- Observation ROVs: small-sized ROVs mainly used for remote observation in shallow waters.
- Light work class ROVs: medium-sized ROVs mostly operating in moderate or deep depths that can manipulate small objects.

- Work class and heavy work class ROVs: large-sized ROVs that can carry many sensors and perform operation in deep depths.

Prototypes are also constantly developed such as the humanoid avatar OceanOne [121], designed for archeology surveys and equipped with haptic hands, or the Dive Buddy ROV developed by the U.S. Navy. Examples of these different ROVs are given in Fig. 1.1.

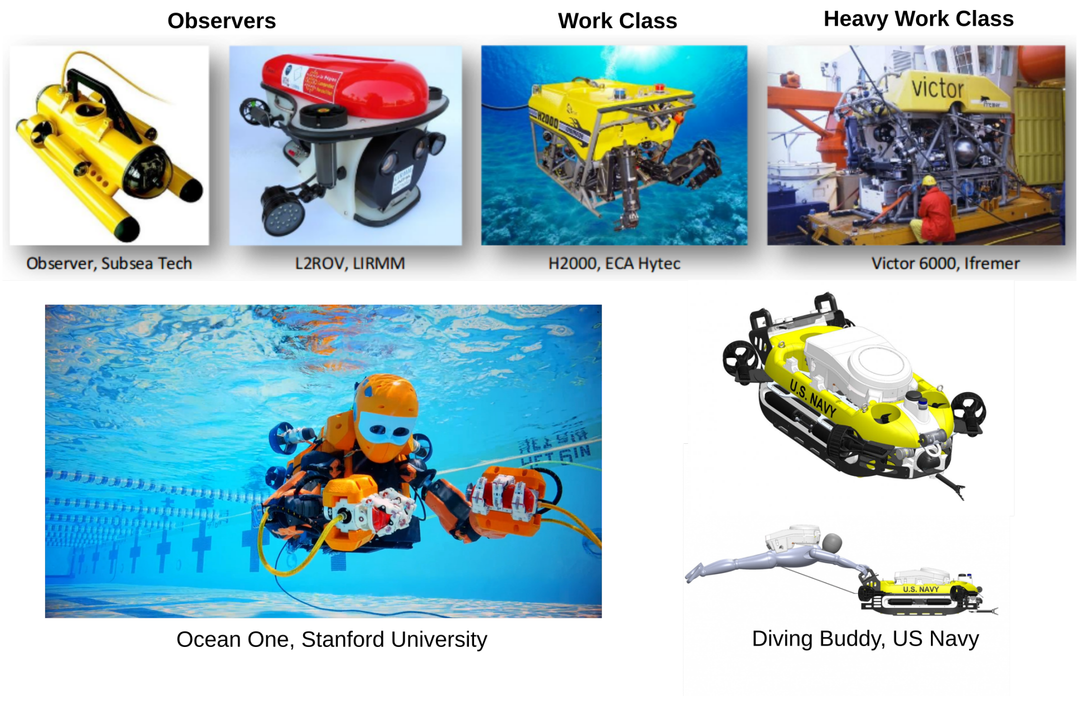


FIGURE 1.1: Example of ROVs from different classes.

For economical reasons, in archaeology, small or medium-sized ROVs are usually used and embed cameras, whose video streams are sent to the surface for piloting and analysis purposes. However, piloting such robots from a video output only is difficult because of the lack of landmarks on the seafloor. Furthermore, the remains of shipwrecks (mostly metallic materials) might create dangerous obstacles, requiring to cautiously navigate in 3D. This is even more important when ROVs are deployed near modern shipwrecks which have become extremely sharp because of corrosion. Therefore, systems providing, in real-time, the localization of the ROVs' are highly beneficial for efficient piloting. Moreover, accurate knowledge of the robot's localization opens the path to many useful applications such as: automatic surveys, servoing or 3D reconstructions.

1.2 The Localization Problem

In robotics, the problem of localization (or *ego*-localization) has been mainly addressed by either dead-reckoning or landmarks-based localization. Dead-reckoning refers to solutions that infer the localization by means of estimating the motions performed by a robot through proprioceptive sensors. On the other hand, landmarks-based solutions rely on exteroceptive sensors to measure some known landmarks in order to infer the current robots' localization.

Simultaneous Localization And Mapping (SLAM) is an in-between solution that tackles the localization problem by building a map of the environment in order to localize the robot within this map. SLAM hence relies on exteroceptive sensing of the robot's environment in order to build such a map. This map then provides landmarks that can be used to estimate the current localization of the robot. When evolving in a 3D environment, the localization is represented by 6 degrees of freedom (DOF), 3DOF to express the position of the robot and 3DOF to express its orientation (or attitude), both with respect to a frame of reference. This 6DOF localization is usually referred to as the pose.

SLAM has been mainly addressed using either LiDARs, SONARs or cameras. Lidars and sonars directly measure the 3D structure of their environment and provide enough information to incrementally build a map at each measurement. Cameras, on the other hand, are bearing-only sensors, that cannot recover the range of their measurements but only provide an information about the direction of the light rays hitting the cameras' sensor. In order to recover a 3D information from cameras' measurements, one may combine them with a vision-based range sensor (RGB-D cameras, Time-Of-Flight cameras). Another solution is to use two cameras, slightly spaced from each other and with an overlapping field of view. This forms a stereoscopic imaging system and distance measurements are obtained by matching pixels corresponding to a same 3D landmark in the images gathered by both cameras. Indeed, if one knows the exact relative poses of each camera with respect to the other, it is then possible to estimate the 3D position of the landmarks by estimating the distance at which the related light rays intercept each other. While these cameras' configuration allows building metric 3D map in unknown environments, SLAM can also be performed using only one camera (referred to as a monocular camera). Similarly to stereoscopic systems, with a monocular camera, one can estimate 3D landmarks by creating a virtual pair of cameras. Such a virtual stereoscopic setup is created by moving the camera and using the images acquired at two different times instead of using simultaneous images from two cameras. Then, if the pose of the monocular camera is known at the images' acquisition time, the estimation of 3D landmarks can be performed in the same way as with stereo cameras. However, in the monocular case, an initialization procedure has to be defined in order to estimate the 3D motion between the two first images that are going to be used to initialize the map. While no map exists, there is no way of estimating the pose of the camera. However, without knowing the poses of the camera at these times, it is not possible to estimate any landmark either. To solve this chicken-and-egg problem, monocular SLAM systems usually rely on multi-view geometry in order to estimate the motion, up to a scale factor, experienced by the camera between the first and the second image. The translation of this motion is then arbitrarily scaled and defines the scale of the map that is going to be built and used for localization. Monocular SLAM methods hence rely on Structure-from-Motion (SfM) techniques in order to solve the localization and mapping problem. Monocular cameras can also be coupled with complementary sensors able to estimate metric motions in order to solve the scale ambiguity and to recover metric estimations.

We next present some sensors and methods that have been used for the specific task of underwater localization.

1.3 Underwater Localization Methods

In this section we first review the most used sensors used to address the localization problem in underwater environments. Then, we propose a review of the state-of-the-art methods developed in the past decades to address the localization and mapping problem for underwater robots.

1.3.1 Localization sensors for underwater environments

We now give a non-exhaustive list of typical sensors used for underwater localization.

1.3.1.1 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is generally composed of a 3-axes gyroscope and a 3-axes accelerometer. Gyroscopes and accelerometers respectively measure angular velocities and linear accelerations. An Inertial Navigation Systems (INS) estimates the position, speed and orientation of an object equipped with an IMU by integrating the IMU measurements. The main issue with an INS is that it accumulates every small errors, leading to significant drift over time. Different technologies have been proposed to manufacture IMUs, with different characteristics in terms of accuracy. The most accurate are often big and bulky in addition to be very expensive. However, even with the finest technology, INS are affected by non-zero drift. In parallel, cheaper sensors exist such as Micro Electro-Mechanical Systems (MEMS) which are strapdown IMUs (*i.e.* with accelerometers and gyroscopes rigidly attached to a platform). MEMS-IMUs have become extremely popular because of both their cheapness and their very small size, which allows to put them in almost any electronic systems. However, such IMUs are quite noisy and are affected by large time-varying biases. Thus, they cannot be directly used in an INS for positioning. Yet, they work pretty well as inclinometers and, if coupled with another sensor able to measure their drift, they can actually give fairly accurate positioning information. Table 1.1 display some of the main characteristics for several IMUs, based on different technologies.

	MPU 9250 [110]	**ADIS 16448** [4]	**EN-1000** [61]
Gyroscope			
Technology	MEMS	MEMS	FOG
Noise (deg / $\sqrt{Hr}$)	0.6	0.66	0.002
Bias Stability (deg / Hr)	> 15.0	14.5	0.01
Accelerometer			
Technology	MEMS	MEMS	MEMS
Bias Stability (m / s^2)	$0.3 \cdot 10^{-3}$	$0.25 \cdot 10^{-3}$	$0.01 \cdot 10^{-3}$
Cross-axis Sensitivity	n/a	< 0.05 deg	< 0.01 deg
Factory Calibrated	No	Yes	Yes
Size ($cm \times cm \times cm$)	$2.3 \times 2.3 \times 0.3$	$2.4 \times 3.8 \times 1.1$	$15.2 \times 9.9 \times 12.4$
Weight (Kg)	< 0.01	0.015	2.03
Price	$\approx 15\$$	$\approx 500\$$	$> 50k\$$

TABLE 1.1: Comparison of different Inertial Measurement Unit technologies.

Gyroscopes

A gyroscope is a sensor that measures the rotation speed (or angular velocity). From a three-axes gyroscope, it is possible to get information in a 3D world. As the measurements provide a velocity information, they have to be integrated in order to get an information about the sensor's current orientation. Gyroscopes are subject to time-evolving biases in addition to classical additive noise measurements. Because of these changing biases, integration of gyroscopes' measurements lead to drift over time. Several technologies exist for such sensors. One of the most accurate one, based on fiber optic gyroscope (FOG), has extremely low bias evolution and even measure the Earth rotation. However, even such gyroscopes drift over time due to the integration of small errors, and they have to be coupled with complementary sensors in order to compensate this drift.

Accelerometers

An accelerometer is a sensor that measures the instantaneous acceleration it is subject to. When static, this sensor measures the inverse of gravity. From a three-axes accelerometer, it is possible to recover the full 3D acceleration of the sensors' carrier. As the gravity axis is observable with static accelerometers, they can be coupled to the gyroscopes' measurements in order to correct the attitude estimations around the axes orthogonal to the gravity axis. Accelerometers are also subject to noise and time-evolving biases. While it is theoretically possible to recover the velocity and position of the sensors' carrier by integration of the acceleration measurements, the noise and evolving biases lead to rapidly increasing error. Furthermore, the acceleration effects due to gravity have to be compensated to get information about velocity and position and even small error in the estimation of the gravity axis will results in error on the velocity and position estimations.

1.3.1.2 Compass

A compass is based on magnetometers and measures the magnetic field of its environment. If placed in an environment with no magnetic disturbance, it measures the Earth's magnetic field and can be used to localize the magnetic north. However, in most cases, magnetic disturbances are present in the sensor environment and a calibration procedure is required to estimate the so-called hard-iron and soft-iron disturbances.

1.3.1.3 Pressure Sensor

An underwater pressure sensor measures the pressure applied by both the atmosphere and the water column above it. As the pressure applied by the water column is linear with respect to its height, the depth from the water surface can be easily computed.

1.3.1.4 Doppler Velocity Logs

A Doppler Velocity Log (DVL) is a sensor that emits four acoustic beams towards static surface or objects and measures the frequency shift of the echoes of these signals. Then, based on the Doppler effect, it provides estimation of the velocity with respect to static objects, such as the sea-floor or the surface.

1.3.1.5 Acoustic Positioning Systems

Acoustic positioning systems can be used as a replacement of GNSS systems in underwater environments. Different technologies exist for such systems and are briefly presented here:

Long Baseline

Long Baseline (LBL) positioning systems use a set of acoustic beacons placed on the sea-floor. By using at least three beacons, and knowing their relative positions, the localization of robots evolving in the area they cover can be estimated by means of time-of-flight measurements and trilateration.

Short Baseline and Ultra-Short Baseline

Short Baseline (SBL) and Ultra-Short Baseline (USBL) sensors combine an array of acoustic transceivers placed on a static object such as a ship hull and a transponder located on the robot. With the array of acoustic transducers, slightly spaced from each other, the position of the robot can be estimated by means of time-of-flight and phase shift measurements. The difference between SBL and USBL mainly lies in the spacing distances between the transceivers, the greater this distance is, the more accurate the positioning will be.

1.3.1.6 LiDARs

A LiDAR sensor is composed of an emitter, projecting laser beams, and a receiver measuring the time-of-flight. In the underwater context, green lasers are used for such sensors, because of their lower absorption.

1.3.1.7 Sonars

Sonars rely on acoustic beams to sense their surrounding environment. There are several types of sonars (echo-sounders, profiling sonars, imaging sonars, side-scan sonars, interferometric sonars…) based on one or many beams and measuring the time-of-flight, acoustic intensity or even the phase shift of the echoes. Detailing all these technologies is out of the scope of this thesis but one can note that, unlike cameras, sonars are not affected by the water's turbidity. However, they provide much less detailed images, at significantly lower rates. Furthermore, tracking features from one sonar image to another is made difficult by the strong dependence of the point of view. Compared to cameras, sonars are more suited to long-range SLAM.

1.3.1.8 Cameras

Cameras are radiometric optical systems that acquire light rays and convert them into numerical values to produce an image. Cameras typically combine a light sensitive sensor with an optical lens. Light rays that goes through the optical lens are projected onto the imaging sensor, which acts as a 2D matrix of Analog to Digital Converters (ADC) to convert the amount of photons received into a numerical image.

1.3.2 SLAM methods for Underwater Robotics

Applying SLAM in an underwater context has been a growing research topic over the past decades. Different SLAM methods, based on different combinations of sensors, have been proposed for navigation, localization, mosaicking or 3D reconstruction. In this section, we review the major SLAM methods that have been proposed in the last twenty years.

In underwater robotics, SLAM has been mainly used as a mean of bounding the drift of the motion estimations produced by fusion of navigational sensors (accelerometer, gyroscope, compass, DVL, pressure sensor) [180]. Localization solutions based on navigational sensors are usually referred to as Dead-Reckoning (DR). In DR systems, the integration of sensors' measurements leads to unrecoverable drift that grows over time. The accumulation of drift is due to both noise in the measurements and linearization errors when estimating pose from sensors with nonlinear measurement models. Incorporating an exteroceptive sensor, such as a sonar or a camera, allows to limit this drift by observing the surrounding environment and finding static landmarks that can be used to estimate the localization of the sensor with respect to these landmarks. If these landmarks can be observed at different times, they will provide non-drifting pose information. Furthermore, if one is able to recognize a place already mapped (*i.e.* a place with already estimated landmarks), the re-observation of previously estimated landmarks will correct the trajectory by decreasing the drift accumulated in-between. The process of recognizing already mapped places is usually referred to as loop-closure detection.

Sonar-based SLAM

Side-scan sonar is one of the preferred sensors for underwater SLAM because of its long range and high resolution. It has first been used with Extended Kalman Filter SLAM (EKF-SLAM) framework in order to update the pose estimations from DR with the sonar measurements [187, 188, 189, 153, 155]. A limitation of EKF-SLAM is the integration of linearization errors in the estimations, which leads to drift over time. In order to limit this drift, Graph SLAM methods have been proposed. Graph SLAM turns the SLAM problem into a factor graph [50] and take into account past estimations when estimating the current pose. The pose estimations are then performed by iteratively minimizing a cost function based on the states to estimate (robot's poses and landmarks) and on the measurements that link them to each other (mainly odometry measurements from DR and landmarks observations). This minimization is performed by modeling the cost function as a nonlinear least-squares problem. While such methods are computationally more demanding than EKF-SLAM, they are also more accurate. This Graph SLAM paradigm have been used with sonars in [115, 69, 214] . In parallel, the use of Particle Filters for representing the 3D map produced by sonars' measurements have been investigated in [12, 11, 13]. While all these methods rely on a Gaussian modeling of the sensors measurements, the use of interval analysis methods has also been proposed in the context of sonar-based SLAM [112, 114, 111, 113, 68, 204, 205].

Vision-based SLAM

The use of cameras has also been greatly investigated for underwater SLAM. Cameras provide a way of estimating 3D landmarks and motion estimations by matching corresponding pixels in overlapping images. While they are subject to the poor imaging conditions of underwater environments, they are also more informative than imaging sonars. As with sonar-based SLAM, cameras have first been used in EKF-SLAM by either incorporating measurements from monocular cameras [222, 8] or stereo cameras [194, 104]. A Visually Augmented Navigation (VAN) framework that uses an Extended Information Filter (EIF), the dual of the EKF, was proposed in [64, 66]. The advantage of this VAN framework is that it can keep past states estimates within the filter, while keeping a low complexity. This VAN framework has both been used with stereo cameras [151, 66] and monocular cameras [65]. However, EIF integrates linearization errors within the filter in the same way as EKF. Therefore, Graph SLAM methods have also been employed with vision-based systems [122, 107, 123]. As cameras provide a way for detecting loop-closures and re-localizing within prior maps, some works proposed to use vision-based systems for multi-session SLAM [178, 176].

Vision-based SLAM in Structured Environments

In case of robots navigating in structured environments, visual localization methods from known patterns have been proposed. In [32, 33, 186, 131], localization is estimated by detecting binary patterns on a pool's floor from a monocular camera. The use of ApriTags [175] with known positions has also been investigated as a mean of removing drift from an EKF-SLAM [118]. In [177], a trinocular system, used in conjunction of navigational sensors in a Graph SLAM, was used to improve the pose estimates when navigating in areas with *a-priori* 3D maps.

Vision-based 3D Reconstruction

The advances in vision-based localization have allowed the development of many underwater 3D reconstruction methods. Vision-based EKF-SLAM have been used to get a prior trajectory before solving the 3D reconstruction problem with the acquired images [203, 182, 116, 223]. The VAN framework has also been used with stereo cameras to produce 3D reconstructions [224, 152, 101]. Some methods proposed also to combine imaging sonars and cameras to produce 3D reconstructions [97, 28]. Graph SLAM methods have also been used to provide prior localization for 3D reconstruction with imaging sonars [219] and cameras [117]. While these methods use the localization estimations of SLAM framework to perform 3D reconstruction, some purely visual methods have been investigated through offline 3D reconstructions using Structure-from-Motion (SfM) techniques [27, 102, 70].

Vision-based Mosaicking

Another great topic of research for underwater localization and mapping has been the use of cameras for mosaicking. In order to create mosaics, images acquired from cameras have to be stitched altogether in order to create visually interpretable 2D maps. The computation of mosaics from homographies has been used as a way of estimating the motions of underwater vehicles using monocular cameras [90, 89, 202]. This method has then been extended to include the measurements of an ultrasonic altimeter in order to recover the metric scale of the motion estimations [91, 88]. These methods have then been enhanced by the use of Kalman Filters in order to improve the localization accuracy [88, 92, 59]. Mosaicking has also been used with vision-based EKF-SLAM [106, 190, 71] that produces more accurate localization estimations than vision-only methods. Recently, the use of interval analysis methods has been investigated for the creation of mosaics [133] and submapping techniques have been successfully used to produce fast mosaics [60]. However, localization and mapping for mosaicking is limited by the assumption of planar scenes. While this assumption is often verified in underwater environments, these methods are not well adapted to scenes with 3D relief. Ortho-mosaicking methods have been proposed to tackle this issue on the mapping side [170, 169]. However, localization cannot be reliably performed in highly 3D environments.

Visual SLAM

Most of the presented solutions can be used for real-time localization of underwater vehicles. However, they are either based on expensive navigational sensors or rely on limiting assumptions about the structure of the environment. A way of localizing in 3D environments from low-cost sensors only is to rely on Visual Odometry (VO) or Visual SLAM (VSLAM) techniques. VO and VSLAM provide solutions for estimating the pose of cameras without any assumption about the explored environment (besides assuming mostly static scenes). VO and VSLAM are very similar in that they estimate the camera's pose by observing 3D landmarks. The main difference between VO and VSLAM lies on the re-observation of previously estimated landmarks in VSLAM, either by means of loop-closure detections or re-tracking of lost landmarks. Stereo VO systems have been proposed in the context of underwater localization in [56, 46, 225, 16, 165] and for real-time photogrammetry [52, 53]. The use of stereo VSLAM has also been investigated in [166, 31, 220, 164]. A few works studied the use of monocular VSLAM and VO. In [23], an EKF is used to perform SLAM with a monocular camera but assumes that the distance to the sea bottom is known and reduce the localization to a 2.5D problem, closer to mosaicking. In [139], a comparison of state-of-the-art monocular VSLAM methods in an underwater context. Localization methods based on the fusion of a monocular camera with a low-cost IMU and a pressure sensor have further been presented in [201, 47].

With respect to all these works, in this thesis we follow the idea of using low-cost sensors for underwater localization. More precisely, we investigate the use of a vision-based system that performs VSLAM from a monocular camera and enhance the pose estimations with a low-cost MEMS-IMU and a pressure sensor. We propose to solve the localization and mapping problem in an archaeological context, where small ROVs might be used. The vision-based system is well adapted to such robots as it only use small sensors, easy to embed and compliant with the limit payload of small robots. Furthermore, recent advances in VSLAM [25] have demonstrated that sub-metric localization can be obtained with such methods. The studies of this thesis have been performed with the idea of providing accurate localization information and 3D mapping capability in real-time.

1.4 Problem Statement

The main objective of this thesis is to rely on a monocular camera for the purpose of localization and mapping in an underwater archaeological context. While monocular setups are not as robust as stereoscopic systems for these tasks, they are easier to embed on a ROV. Moreover, they do not have any synchronization issue and many underwater housings are well fitted to their shape. It is also easier to set monocular cameras behind spherical domes, that reduces the diffraction effects, than stereoscopic ones. Furthermore, the scale issue of monocular systems can be solved by adding complementary sensors. In this thesis, we investigate the use of a low-cost MEMS-IMU and a pressure sensor as additional sensors. The great advantage of this setup is that it is self-contained, in the sense that every sensors can fit in a single housing that provides everything required for running a localization and mapping algorithm.

The different problems that we address in this thesis are the following:

- How to efficiently process video streams of underwater images for the purpose of localization and mapping ?
- How to accurately ego-localize a robot with only a monocular camera in underwater environments ?
- How to optimally fuse a pressure sensor and a MEMS-IMU with a monocular camera for improved localization ?
- How to get a dense 3D reconstruction from a monocular based setup ?

As these problems have been addressed with an out-of-lab application target, the technical challenges of real-time processing and hardware design have been taken into consideration.

1.5 Thesis contributions

In this thesis, we investigate the use of vision-based SLAM to tackle the underwater localization challenge in an archeological context. More specifically, we first study the efficiency of features tracking methods on images that are affected by typical underwater visual disturbances. Then, we propose a monocular Visual SLAM algorithm that is well suited to underwater environments. This monocular SLAM method is then extended to include the measurements of a pressure sensor and of a low-cost MEMS-IMU. The resulting Visual-Inertial-Pressure SLAM method integrates the different sensors' measurements by means of tight fusion within the Visual SLAM algorithm and provides scaled estimations. Two prototypes of acquisition systems composed of a monocular camera, a pressure sensor, a low-cost MEMS-IMU and an embedded computer have been developed for field experiments. We have also used these acquisition systems to record a large dataset, that has been publicly released. Furthermore, the developed SLAM method has been implemented on the embedded computer and has been successfully tested in real-time. Finally, we propose a monocular dense 3D reconstruction method that works online.

1.6 Thesis Outline

The thesis is organized as follows:

- ***Chapter 2*** details the mathematical tools that will be used within this thesis.
- ***Chapter 3*** presents the conducted evaluation of different features tracking methods on underwater images with typical visual disturbances.
- ***Chapter 4*** details the monocular Visual SLAM method developed for the context of underwater archaeology.
- ***Chapter 5*** presents an extension of the monocular SLAM method to integrate other sensors. We first propose a Visual-Pressure SLAM method that integrates the measurements of a pressure sensor in order to increase the accuracy and recover the metric scale of the estimated trajectories. The method is further extended to include the measurements of a MEMS-IMU to increase the robustness of the SLAM system.
- ***Chapter 6*** describes the design of the acquisition systems and introduces the dataset used to evaluate the SLAM methods described in the previous chapters. It further presents the proposed dense 3D reconstruction method and results obtained on the acquired dataset.

Videos

For the interested readers, videos of the proposed SLAM and 3D reconstruction methods in action are available at :
https://www.youtube.com/channel/UCFsvlI143Evf2F2sF5Hbxuw

Publications

At the time of writing, the following articles have been published in relation within the work presented in this thesis:

Journal Papers

Maxime Ferrera, Vincent Creuze, Julien Moras, and Pauline Trouvé-Peloux. "AQUALOC: An Underwater Dataset for Visual-Inertial-Pressure Localization." In: *The International Journal of Robotics Research*. 2019

Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Real-Time Monocular Visual Odometry for Turbid and Dynamic Underwater Environments". In: *Sensors*. Vol. 19. 3. 2019

International Conference Papers

Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, Vincent Creuze, and Denis Dégez. "The Aqualoc Dataset: Towards Real-Time Underwater Localization from a Visual-Inertial-Pressure Acquisition System". In: *IROS Workshop - New Horizons for Underwater Intervention Missions: from Current Technologies to Future Applications*. 2018

National Conference Papers

Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Odométrie Visuelle Monoculaire en Environnement Sous-Marin". In: *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP)*. 2018

Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Localisation autonome basée vision d'un robot sous-marin et cartographie de précision". In: *ORASIS*. 2017

Chapter 2

Mathematical Notions

Contents

2.1 Introduction

This chapter presents most of the mathematical tools that will be used within this thesis. We first review the geometry of a camera and the associated multi-view geometry. Then, we present the optimization tools that are going to be used for the task of state estimations and sensors' measurements fusion. We will go through the notions related to Least-Squares, Maximum Likelihood Estimation and Factor Graphs. Finally, we will present the Lie Groups' properties that apply to 6DOF pose states.

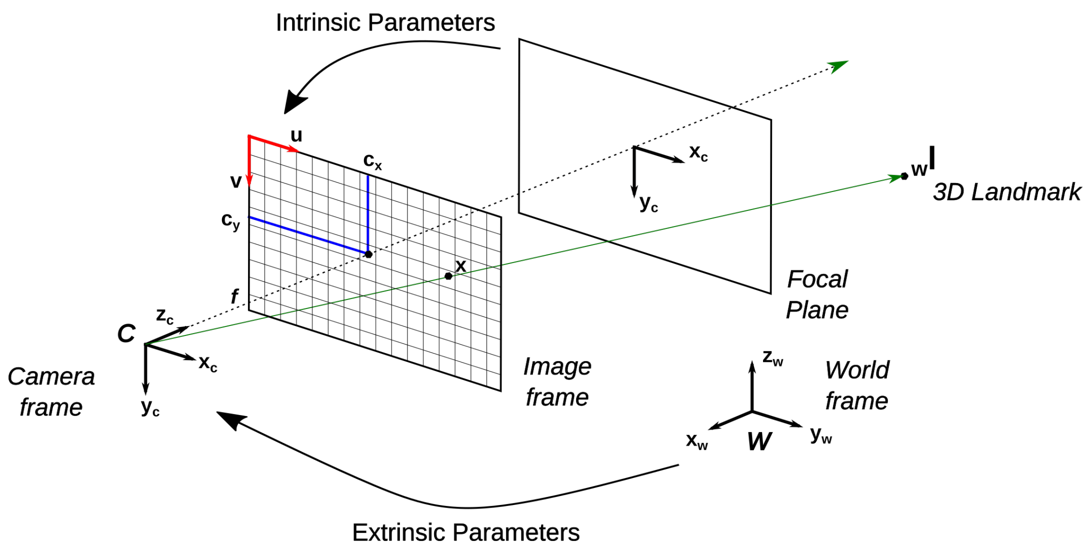


FIGURE 2.1: Pinhole camera model scheme.

An in-depth presentation of each fields is out of the scope of this thesis, instead we give an overview of the specific tools and notions that we will use in the next chapters. However, for the interested readers, we give references to more detailed materials about each topic.

2.2 Pinhole Camera Model

The pinhole camera model is the most classical geometric model of the image formation process for a camera [98] and is depicted into Fig. 2.1. This model considers that the pose of the camera is expressed at the optical center (*i.e.* at the pinhole), with its z-axis oriented towards the imaged scene, its x-axis oriented along the horizontal axis of the produced 2D image and pointing towards the right border of the image and its y-axis oriented along the vertical axis of the 2D image and pointing towards the down border. The extrinsic parameters of the camera express the transformation from a frame of reference w (often referred to as the world frame) to the camera frame c. In a 3D context, the pose of the camera is defined by 6DOF. Mathematically, the pose is expressed in the world frame by a rotation matrix $\mathbf{R}_{wc} \in \mathbb{SO}(3)$ and a translation vector $\mathbf{t}_{wc} \in \mathbb{R}(3)$, where $\mathbb{SO}(3)$ is the Special Orthogonal group that defines the behavior of rotation matrix. Cameras' poses are hence elements of the Special Euclidean group $\mathbb{SE}(3)$. The transformation from the camera frame to the world frame $\mathbf{T}_{wc} \in \mathbb{SE}(3)$ can be written in the form of a homogeneous matrix:

$$\mathbf{T}_{wc} = \begin{bmatrix} \mathbf{R}_{wc} & \mathbf{t}_{wc} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.1}$$

And the inverse transformation, $\mathbf{T}_{cw} \in \mathbb{SE}(3)$, from the world frame to the camera frame is defined by the inverse homogeneous matrix:

$$\mathbf{T}_{cw} = \mathbf{T}_{wc}^{-1} = \begin{bmatrix} \mathbf{R}_{wc}^{T} & -\mathbf{R}_{wc}^{T} \cdot \mathbf{t}_{wc} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{cw} & \mathbf{t}_{cw} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.2}$$

The transformation $\mathbf{T}_{cw}$ corresponds to the extrinsic parameters of the camera. A calibrated camera is also characterized by intrinsic parameters that represents the image formation model. The focal length of a camera's lens defines a virtual 2D plane which intercepts every rays coming from 3D objects in the world frame and projecting onto the camera's imaging sensor. This imaging sensor is composed of 2D matrix of light sensitive receptors that convert the amount of receive light into a numerical value. In case of a monochromatic camera, these values are directly the pixel intensity values. The imaging sensor hence defines a 2D plane whose coordinates are expressed in pixels and which originates at the most top-left pixel. The intrinsic parameters represent the coefficients that convert the position of points in the focal plane frame, into their pixel projection on the image frame. In order to take into account the fact that any point along a defined ray passing through the camera's optical center will project onto the same pixel, the intrinsic parameters are usually expressed as a homogeneous matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ whose most general formulation is:

$$\mathbf{K} = \begin{bmatrix} k_u \cdot f & s & c_x \\ 0 & k_v \cdot f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where k_u and k_v are scaling factors in the horizontal and vertical image axes, respectively, and represent the possible non squareness of the pixels, s is a skew factor that can be used to model any shear distortion (non rectangular pixels) in the image formation process and c_x and c_y are the pixel coordinates of the principal point of the image, that is the coordinates of the pixel that receives the light ray perfectly collinear with the camera's z-axis.

For modern cameras, the skew factor is often not significant and the intrinsic matrix $\mathbf{K}$ can be written as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

where f_x and f_y directly encodes the focal length effects in terms of pixels. This intrinsic matrix can be estimated by means of a calibration procedure [228, 229] using an object with known dimensions (typically a checkerboard or an apritlag). The matrix $\mathbf{K}$ is thus often referred to as the intrinsic calibration matrix.

The mathematical projection model of a point from the world frame into the image frame of a camera is finally defined in terms of both the intrinsic and extrinsic parameters:

$$\mathbf{x}' = \mathbf{K} \cdot \begin{bmatrix} \mathbf{R}_{cw} & \mathbf{t}_{cw} \end{bmatrix} \cdot {}_w\mathbf{l}' \tag{2.5}$$

where ${}_w\mathbf{l}' \in \mathbb{P}^3$ is the homogeneous position of a 3D landmark in the world frame and $\mathbf{x}' \in \mathbb{P}^2$ is the homogeneous pixel coordinate of the projection of ${}_w\mathbf{l}'$ in the image frame. Because of the projective geometry of the pinhole camera, the homogeneous form is often employed to denote that any point along a ray that goes through the optical center of a camera will be projected onto the same pixel. Furthermore, the homogeneous expressions are very useful for representing the projective transformations in terms of linear algebra (*i.e.* in terms of matrices and vectors multiplication only).

In order to recover the Cartesian coordinates ${}_w\mathbf{l} \in \mathbb{R}^3$ and $\mathbf{x} \in \mathbb{R}^2$ from ${}_w\mathbf{l}'$ and $\mathbf{x}'$, we need to remove their projective factors λ_l and λ_x:

$${}_w\mathbf{l}' = \begin{bmatrix} x_w \\ y_w \\ z_w \\ \lambda_l \end{bmatrix} = \frac{1}{\lambda_l} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \longrightarrow {}_w\mathbf{l} = \frac{1}{\lambda_l} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \tag{2.6}$$

$$\mathbf{x}' = \begin{bmatrix} u \\ v \\ \lambda_x \end{bmatrix} = \frac{1}{\lambda_x} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \longrightarrow \mathbf{x} = \frac{1}{\lambda_x} \cdot \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.7}$$

Note that homogeneous coordinates can also be used to represent *vanishing* points (also called points at infinity) by setting the projective factor to zero. The inverse conversion, from Cartesian coordinates to homogeneous coordinates consists merely in adding one dimension to the Cartesian coordinates at setting its value to 1 (*i.e.* adding a projective factor set to 1).

This projection model can also be expressed in terms of Cartesian coordinates through the projection function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ that projects point in the camera frame in the image frame:

$$\mathbf{x} = \pi\left(\mathbf{K} \cdot_c \mathbf{l}\right) \quad \text{and} \quad {}_c\mathbf{l} = \mathbf{R}_{cw} \cdot_w \mathbf{l} + \mathbf{t}_{cw} \tag{2.8}$$

$$\mathbf{x} = \begin{bmatrix} \frac{x_c}{z_c} \cdot f_x + c_x \\ \frac{y_c}{z_c} \cdot f_y + c_y \end{bmatrix} \quad \text{with} \quad {}_c\mathbf{l} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{2.9}$$

The projection model presented here is yet only valid for undistorted images. In reality, because of lens' imperfections, the images produced by a camera are distorted. The distortion effects can be modeled by means of a polynomial in order to define a bijective mapping between distorted and undistorted coordinates. The coefficients of the distortion polynomial can be estimated during the calibration procedure. We refer the readers to [213] for more information about camera and distortion models. In the rest of this thesis we will always consider distortion free images unless when explicitly stating it.

2.3 Multi-View Geometry

In this section we review some methods from the multi-view geometry theory. Here, we mainly focus on the tools that will be used within this thesis. We refer the interested reader to [98] for an in-depth of the whole multi-view geometry field.

2.3.1 Triangulation

Triangulation refers to the estimation of a 3D point from its observation in two cameras whose poses are known. The estimation of the 3D point is performed by finding the point of intersection of the rays projected from this point in both cameras. However, because of noise in the measurements (*i.e.* non perfect observations of the projected 3D point), the two rays never intersect for real. The triangulation problem is hence solved by searching for the most likely intersection of two rays if theirs observations were noiseless.

Several methods have been proposed to solve the triangulation problem. The mid-point method approximates the intersection of the rays as the middle of the shortest segment that can be drawn between the two rays. A linear

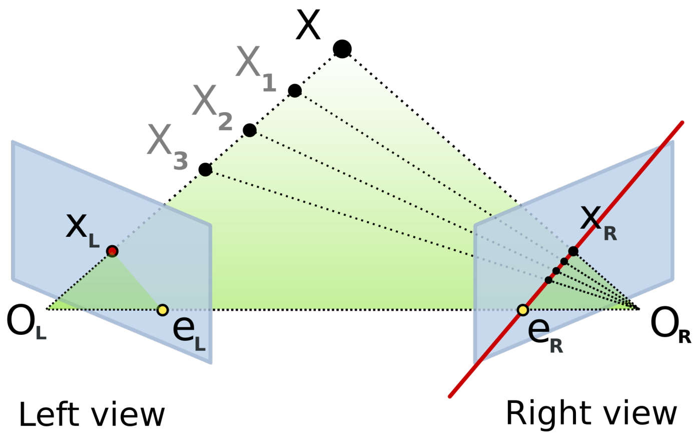


FIGURE 2.2: Epipolar geometry properties example (taken from https://en.wikipedia.org/wiki/Epipolar_geometry/).

method based on the Direct Linear Transform (DLT) turns the triangulation problem in a linear system of equations which is solved by means of SVD (Singular Value Decomposition). A nonlinear closed-form approximation method has also been proposed for fast triangulation.

Details about these different approaches to the triangulation problem can be found in [99, 119].

2.3.2 Epipolar Geometry

The projective geometry that defines the mathematical model of a perspective camera leads to some geometrical rules about the formation of images of a same scene taken from different view points. For instance, when considering images acquired by two cameras (*i.e.* by a stereo vision system), the multi-view geometry properties are those of epipolar geometry. Epipolar geometry defines the rules that apply on the projection of 3D rays into two images of a same scene, taken from two distinct positions.

The epipolar geometry properties mostly define point-to-line relationships that have to hold between the projections of 3D points into two images. As shown on Fig. 2.2, any point X_L in the left image is the projection of a 3D point X that goes through the left camera optical center. Even if the range of the ray that is defined by the projection of X in the left image is not known, epipolar geometry shows that this ray has to appear along a specific line on the right image, the epipolar line. Epipolar lines are defined by the epipoles, that is the projection of the cameras' optical centers into their respective frames. Therefore, an epipolar plane can be defined for any 3D point that projects in both images from the position of this 3D point, the optical centers of both cameras and their resulting epipoles.

Epipolar geometry can be used to define 2D / 2D constraints between points in the two images. Given two corresponding points $\mathbf{x}'_L$ and $\mathbf{x}'_R$ in homogeneous coordinates, respectively defined in the left and right image frames, epipolar geometry tells us that there exists a matrix $\mathbf{F} \in \mathbb{R}^{3\times 3}$ such that:

$$\mathbf{x}'^T_L \cdot \mathbf{F} \cdot \mathbf{x}'_R = 0 \tag{2.10}$$

where $\mathbf{F} \cdot \mathbf{x}'_R$ defines the epipolar line in the left camera associated to the 3D ray observed by $\mathbf{x}'_R$ in the right image. The matrix $\mathbf{F}$ is called the Fundamental Matrix.

As epipolar geometry depends on the relative pose of the stereo vision setup, the Fundamental Matrix encodes the relative transformation that exists between both cameras, up to a scale factor. Given $\mathbf{K}_L$ and $\mathbf{K}_R$, the intrinsic matrices of the left and right cameras, and $\mathbf{R}$ and $\mathbf{t}$ the rotation matrix and translation vector that transforms any point expressed in the right camera frame into the left camera frame, the Fundamental Matrix $\mathbf{F}$ can be written as:

$$\mathbf{F} = (\mathbf{K}_L \cdot \mathbf{R}_L)^{-T} \cdot (\mathbf{t}_R - \mathbf{t}_L)^{\wedge} \cdot \left(\mathbf{R}_R^T \cdot \mathbf{K}_R^{-1}\right) \tag{2.11}$$

where $(\cdot)^{\wedge}$ is the skew matrix operator that turns a 3D vector into a skew matrix.

Any Fundamental Matrix is of rank 2 and can be estimated from 8 corresponding points [147]. Furthermore, the Fundamental Matrix also encodes the intrinsic parameters of both cameras and can hence be used with uncalibrated cameras.

In the case of calibrated cameras, the Essential Matrix $\mathbf{E} \in \mathbb{R}^{3\times 3}$ can be directly used. The Essential Matrix is defined as follow:

$$\mathbf{E} = \mathbf{K}_L^{-1} \cdot \mathbf{F} \cdot \mathbf{K}_R \tag{2.12}$$

where $\mathbf{E}$ is a specialization of $\mathbf{F}$ in the specific case of having normalized image coordinates (*i.e.* coordinates obtained by removing the effect of the calibration matrix $\mathbf{K}$), such that:

$$\tilde{\mathbf{x}}_L^T \cdot \mathbf{E} \cdot \tilde{\mathbf{x}}_R = 0 \tag{2.13}$$

$$\text{with } \tilde{\mathbf{x}}_L^T = \mathbf{K}_L^{-1} \cdot \mathbf{x'}_L^T \text{ and } \tilde{\mathbf{x}}_R^T = \mathbf{K}_R^{-1} \cdot \mathbf{x'}_R^T \tag{2.14}$$

The Essential Matrix hence only depends on the relative pose of the stereo vision setup and has fewer degrees of freedom than the Fundamental Matrix. Any Essential Matrix must have two equal singular values and one null singular value and can therefore be estimated from 5 corresponding points [172]. The relative pose, up to a scale factor on the translation vector, can be recovered by means of SVD.

As the estimation of Essential Matrices requires less corresponding points than Fundamental Matrices, they are to be preferred when working with calibrated cameras. Furthermore, the Fundamental Matrix is known to produce false results when estimated from coplanar points because of a degenerate configuration whereas the Essential Matrix is more robust to such configurations [172].

2.3.3 Pose Estimation

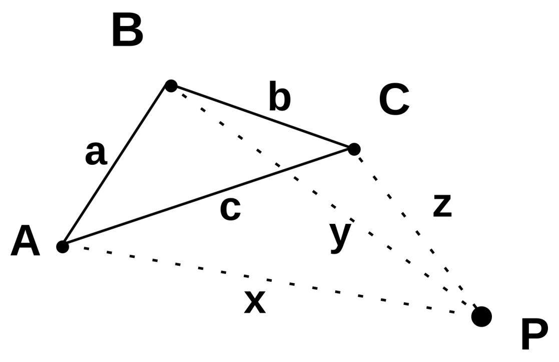


FIGURE 2.3: Illustration of the P3P problem. P represents the optical center of the camera and A, B, C represent 3 known points.

In the case of calibrated cameras, the pose of the camera with respect to the world frame w can be estimated from the observation of at least three known 3D world landmarks. More exactly, a set of hypotheses can be extracted from three 2D / 3D correspondences and the correct pose can be recovered from these hypotheses by disambiguating with an additional 2D / 3D correspondence. When using this minimal configuration of three correspondences, the visual pose estimation problem is referred to as *Perspective-from-3-points* (P3P). The P3P problem, illustrated in Fig. 2.3, consists in recovering the 6DOF pose of the camera by solving the following system of equations:

$$\begin{cases} x^2 + z^2 - y \cdot z \cdot p - b^2 = 0 \\ z^2 + x^2 - x \cdot z \cdot q - c^2 = 0 \\ x^2 + y^2 - x \cdot y \cdot r - a^2 = 0 \end{cases} \tag{2.15}$$

with $x = |AP|$, $y = |BP|$, $z = |CP|$, $a = |AB|$, $b = |BC|$, $c = |AC|$, $\alpha = \angle BPC$, $\beta = \angle APC$, $\gamma = \angle APB$, $p = 2\cos(\alpha)$, $q = 2\cos(\beta)$ and $r = 2\cos(\gamma)$.

By solving Eq.(2.15) for x, y, z, α, β and γ, one can recover the 6DOF of the camera pose whose optical center is P. Several algorithms have been proposed over the years to solve this system [86, 127, 120].

All the available 2D / 3D correspondences in an image can also be used altogether to estimate more robustly the pose of the camera. In this case, the visual pose estimation problem is referred to as *Perspective-from-n-points* (PnP). The PnP problem can be solved by finding an initial solution with the P3P method and then turn the PnP problem in the form of a least-squares optimization. Some algorithms have also been proposed to directly estimate a pose solution from more than four correspondences [135, 129].

2.3.4 Robust Estimation with RANSAC

In order to use the set of formulas provided by the theory of multi-view geometry, correspondences have to be found. A first step of data association is hence required to provide 2D / 2D or 2D / 3D correspondences. However, in computer vision, data association will never be perfect and some wrong correspondences (often referred to as outliers) will appear. Furthermore, because of noise in the visual measurements, there will be no exact solution when solving multi-view problems and, as there are usually more correspondences than the minimal amount required, not all configurations will return the same result.

One way of copping with this data discrepancy is to rely on a RANdom SAmple Consensus (RANSAC) scheme [78]. The idea of RANSAC is to rely on a voting consensus to estimate a set of parameters. This is done by iteratively selecting a minimal set of correspondences, estimating the parameters searched for from this set and counting the number of correspondences that agree with the current estimation. In order to perform the counting, an error tolerance threshold is defined and, for every candidate, the resulting error given the current estimated parameters is used to vote for this set of parameters or not. This process is continued until a set of parameters with enough votes is found or the maximum number of allowed iterations is reached. The optimal maximum number of iterations to find a good solution can be computed and depends on the expected ratio of outliers within the data, the minimum number of samples required to perform one estimation and the probability of success required.

The effects of RANSAC are twofolds. First, it returns the most likely solution in terms of consensus as it keeps the solution the most data agrees upon. Second, it provides a way of detecting possible outliers within the data. Indeed, as outliers will not fulfill the geometrical properties of multi-view geometry, data that do not agree with the selected solution can be considered as unreliable and discarded.

The use of RANSAC provides a way of performing robust estimations from data corrupted by outliers. Furthermore, as it also returns the sets of inliers and outliers, it can be used to remove as soon as possible any wrong data association. This is even more useful if one wants to refine the estimated parameters with all the available data (*i.e.* solving an overdetermined system with noisy data). This is usually done by means of least-squares estimations, which are very sensitive to the presence of outliers within the data. However, when coupled with a RANSAC pre-processing, least-squares estimations can be performed on the inliers only.

2.4 Least-squares methods

In the context of computer vision and multi-view geometry, the number of data associations (or measurements in a more general way) is almost always far greater than the minimum number of samples required for the different problems to solve. As these measurements are noisy, there is no exact solution and the problem is overdetermined.

In order to solve these overdetermined problems, least-squares methods can be applied [173]. Least-squares methods perform an estimation of a set of parameters χ by minimization of the sum of squared errors between the measurements and the model:

$$L(\chi) = \sum_i \|f_i(\chi)\|^2 \tag{2.16}$$

where $L(\chi)$ is the sum of squared errors and each $f_i(\cdot)$ denotes the residual functions associated to each measurement. The sum term can be removed by stacking all the residuals $f_i(\cdot)$ into a vector $\mathbf{f}$:

$$L(\chi) = \|\mathbf{f}(\chi)\|^2 \tag{2.17}$$

For linear measurement models, the linear least-squares problem to solve is the following:

$$\chi^* = \underset{\chi}{\arg\min}\, L(\chi) = \underset{\chi}{\arg\min}\, \|\mathbf{b} - \mathbf{H} \cdot \chi\|_2^2 \tag{2.18}$$

where $\mathbf{H}$ is the observation model matrix and $\mathbf{b}$ is the set of measurements.

If $\mathbf{H}$ is positive definite, the parameter vector χ is then estimated by solving the normal equations:

$$\chi = \left(\mathbf{H}^T \cdot \mathbf{H}\right)^{-1} \cdot \mathbf{H}^T \cdot \mathbf{b} \tag{2.19}$$

If the measurements are modeled as corrupted by gaussian noise, the weighted least-squares method can be used to weight the influence of every measurement using their covariances $\mathbf{\Sigma}$. In this case, the minimization is performed over the sum of squared Mahalanobis distances:

$$\chi = \underset{\chi}{\arg\min}\, \|\mathbf{b} - \mathbf{H} \cdot \chi\|_{\mathbf{\Sigma}}^2 \tag{2.20}$$

$$\chi = \left(\mathbf{H}^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{H}\right)^{-1} \cdot \mathbf{H}^T \cdot \mathbf{\Sigma}^{-1}\mathbf{b} \tag{2.21}$$

For nonlinear measurement models however, the least-squares minimization cannot be performed in one step and optimization methods have to be used.

Let $h(\cdot)$ be the nonlinear function that defines the observation model, the weighted nonlinear least-squares cost function is of the form:

$$L(\chi) = \|\mathbf{b} - h(\chi)\|_{\mathbf{\Sigma}}^2 \tag{2.22}$$

This cost function has to be optimized in order to find the best estimates for χ.

2.4.1 Gauss-Newton

A nonlinear least-squares problem can be optimized with the Gauss-Newton method. This method linearizes the function $h(\cdot)$ around the current estimate of χ in order to solve the nonlinear least-squares problem. The linearization is done by a first-order Taylor expansion:

$$h(\chi + \delta\chi) \approx h(\chi) + \frac{\partial h}{\partial \chi} \cdot \delta\chi \tag{2.23}$$

The matrix $\mathbf{J}_\chi = \frac{\partial h}{\partial \chi}$ is the Jacobian of the cost function. Linearizing the cost function thus results in the following approximation:

$$L(\chi + \delta\chi) = \|\mathbf{f}(\chi + \delta\chi)\|^2_\Sigma = \|\mathbf{b} - h(\chi + \delta\chi)\|^2_\Sigma \tag{2.24}$$

$$L(\chi + \delta\chi) \approx (\mathbf{e}(\chi) - \mathbf{J}_\chi \cdot \delta\chi)^T \cdot \mathbf{\Sigma}^{-1} \cdot (\mathbf{e}(\chi) - \mathbf{J}_\chi \cdot \delta\chi) \tag{2.25}$$

where $\mathbf{e}(\chi) = \mathbf{b} - h(\chi)$ is the residual vector. The Gauss-Newton optimization search for updates $\delta\chi$ that minimize the cost function L. The linearization fixes the current state estimates in order to compute the residuals associated to the cost function and turns the nonlinear problem into a linear one. The optimal update is searched along the local gradient of the cost function, defined by the Jacobian, and its search direction is conditioned by the Hessian of the cost function, approximated by $\mathbf{J}_\chi^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{J}_\chi$ in this case.

By deriving the cost function L with respect to $\delta\chi$ and setting to zero, one obtains the following normal equations:

$$\delta\chi = \left(\mathbf{J}_\chi^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{J}_\chi\right)^{-1} \cdot \mathbf{J}_\chi^T \cdot \mathbf{\Sigma}^{-1}\mathbf{e}(\chi) \tag{2.26}$$

Note that the normal equations are exactly the same as in the linear least-squares case. This is because the Gauss-Newton method approximates the nonlinear least-squares problem as a linear one around a current guess.

The minimization problem is built and solved iteratively, starting from an initial guess about the set of parameters χ. After each iteration, the parameters are updated with the solution to the normal equations: $\chi^* = \chi + \delta\chi$. The updated parameters χ^* are then used at the next iteration to re-compute the Jacobian and residuals.

Note that because of the nonlinearity and non-convexity of most of the computer vision problems, the accuracy of the initial guess about the parameters χ is crucial to obtain good results and not fall in local minima.

2.4.2 Levenberg-Marquardt

The Levenberg-Marquardt method is a trust-region extension of the Gauss-Newton method that better ensures convergence. The minimization of a non-linear cost function is conditioned by the provided initial solution (*i.e.* the initial values of χ). When far from the optimum, the Gauss-Newton optimization might leads to updates that increase the value of the cost function or get stuck into local saddle points.

This method proposes to enhance the normal equations with a damping factor α:

$$\delta\chi = \left(\underbrace{\mathbf{J}_\chi^T \cdot \mathbf{\Sigma}^{-1} \cdot \mathbf{J}_\chi}_{\mathbf{A}} + \alpha \cdot \mathrm{diag}(\mathbf{A}) \right)^{-1} \cdot \mathbf{J}_\chi^T \cdot \mathbf{\Sigma}^{-1} \mathbf{e}(\chi) \tag{2.27}$$

where $\mathbf{A}$ is the approximated Hessian of the cost function. The damping factor α acts as a weighting factor which turns the optimization to be performed like a gradient descent for large α and like the Gauss-Newton approach for small α. This damping factor is updated at each iteration by decreasing it if the parameters' update has led to a significant decrease of the cost function and by increasing it otherwise. In cases where the cost function would not decrease by a significant amount, the parameters' update is not applied and a new update is computed by again solving Eq.(2.27) for an increased value of α.

2.5 Maximum Likelihood Estimation

In a Bayesian framework, any sensors measurements are random samples from a specific probabilistic distribution. State estimation in robotics can benefit from this modeling as it provides theoretical tools on how to efficiently fuse multiple sensors' measurements, with different modalities and noise characteristics.

If the state vector χ is modeled as a multi-variate random variable that follows a normal distribution, χ can be expressed as:

$$\chi \sim \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{2.28}$$

where $\mu \in \mathbb{R}^n$ is the mean, $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is the covariance and $n = \mathbf{dim}(\chi)$.

The sensors measurements $\mathbf{z}_i$ can further be modeled as being corrupted by zero-mean gaussian noise η:

$$\mathbf{z}_i = h(\chi) + \eta \quad \text{with} \quad \eta \sim \mathcal{N}(\mathbf{0}, \Sigma_i) \tag{2.29}$$

where $h(\cdot)$ is the observation model function. The measurements can thus be defined in terms of conditional probability $p(\mathbf{z_i}|\chi)$:

$$p(\mathbf{z_i}|\chi) = \frac{1}{\sqrt{(2\pi)^n \cdot \det \Sigma_i}} \cdot \exp\left(-\frac{1}{2}\left(\mathbf{z}_i - h(\chi)\right)^T \cdot \Sigma_i^{-1} \cdot \left(\mathbf{z}_i - h(\chi)\right)\right) \tag{2.30}$$

which can be simplified by dropping the normalization term:

$$p(\mathbf{z_i}|\chi) \propto \exp\left(-\frac{1}{2}\left(\mathbf{z}_i - h(\chi)\right)^T \cdot \Sigma_i^{-1} \cdot \left(\mathbf{z}_i - h(\chi)\right)\right) \tag{2.31}$$

$$p(\mathbf{z_i}|\chi) = \exp\left(-\frac{1}{2}\left\|\mathbf{z}_i - h(\chi)\right\|^2_{\Sigma_i}\right) \tag{2.32}$$

where $\|\mathbf{z}_i - h(\chi))\ \|^2_{\Sigma_i}$ is the squared Mahalanobis distance.

Given a set of measurements $\mathbf{z}$, the state estimation problem can be expressed with the Bayes rule:

$$p(\chi|\mathbf{z}) = \frac{p(\mathbf{z}|\chi) \cdot p(\chi)}{p(\mathbf{z})} \tag{2.33}$$

where $p(\chi)$ is the *prior* over the state vector, $p(\mathbf{z})$ is the normalized probability factor and $p(\mathbf{z}|\chi)$ denotes the conditional probability over the set of measurements.

Considering the measurements $\mathbf{z}_i$ as being independent, $p(\mathbf{z}|\chi)$ can be written as the following product:

$$p(\mathbf{z}|\chi) = \prod_i p(\mathbf{z}_i|\chi) \tag{2.34}$$

Therefore, from the Bayes equation (2.33), we can infer the states of χ given the measurements $\mathbf{z}$ with a Maximum A-Priori (MAP) estimator:

$$\begin{aligned} \chi^* &= \arg\max_{\chi} p(\chi|\mathbf{z}) \\ &= \arg\max_{\chi} \frac{p(\mathbf{z}|\chi) \cdot p(\chi)}{p(\mathbf{z})} \end{aligned} \tag{2.35}$$

As χ does not depend on $p(\mathbf{z})$, this term can be dropped:

$$\chi^* = \arg\max_{\chi} p(\mathbf{z}|\chi) \cdot p(\chi) \tag{2.36}$$

Furthermore, in the case where we have no *a-priori* information about χ, Eq.(2.35) can be turned in the form of a Maximum Likelihood Estimation (MLE):

$$\chi^* = \arg\max_{\chi} p(\mathbf{z}|\chi) \tag{2.37}$$

The MLE returns the state vector that maximizes the joint probability of $p(\mathbf{z}|\chi)$. In other words, the MLE searches for the state vector that will maximize the probability of having the measurements $\mathbf{z}$.

Taking the negative logarithm of equation (2.37) and recalling Eq.(2.32), one obtain:

$$\chi^* = \arg\min_{\chi} -\frac{1}{2}\sum_i \|\mathbf{z}_i - h(\chi)\|^2_{\Sigma_i} \propto \arg\min_{\chi} \sum_i \|\mathbf{z}_i - h(\chi)\|^2_{\Sigma_i} \tag{2.38}$$

Infering a vector of state from a set of measurements through MLE is done, in the context of gaussian modeling, by solving the least-squares problem defined by $\sum_i \|\mathbf{z}_i - h(\chi)\|^2_{\Sigma_i}$. The MLE can hence be solved with the Gauss-Newton or Levenberg-Marquardt algorithms presented in the previous section.

2.6 Factor Graphs

Factor graphs are a useful tool for defining a state estimation problem and representing the underlying probability density functions. An in-depth review of the use of factor graphs for robotics state estimation is given in [50].

In a factor graph, states are represented as nodes and these nodes are linked by factors (or edges), representing the probability density function associated to some measurements. Considering the simple problem in Fig. 2.4, with five nodes (three pose states and two landmarks) where the pose state X_1 and X_2 are linked to the landmark l_1 and the pose states X_2 and X_3 are linked to the landmark l_2, we can write the resulting likelihood probability as:

$$\begin{aligned} p(\mathbf{z} = \{z_{ij}\} | \chi = \{l_i, X_j\}) = p(z_{11}|X_1, l_1) \cdot p(z_{12}|X_2, l_1) \\ \cdot p(z_{22}|X_2, l_2) \cdot p(z_{23}|X_3, l_3) \end{aligned} \tag{2.39}$$

The Maximum Likelihood Estimate of the problem can hence be derived seamlessly from the factor graph representation. Furthermore, additional

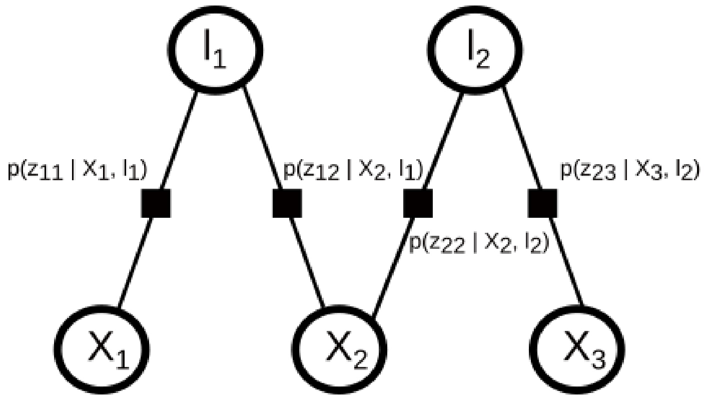


FIGURE 2.4: Toy factor graph example. The five nodes represent the states to estimate, with X_i representing pose states and l_j landmark states. The factors define the probability density functions associated to the measurements z_{ij}.

sensors measurements, with different modalities, can be easily inserted in such graphs. Even priors on states, or direct state measurements, can be smoothly added through the use of unary edges (*i.e.* edges connected to just one state).

Factor graphs are therefore very useful to represent the relationship between states and measurements for the purposes of state estimation and optimization. They are a very common way for defining optimization problem with many states and many measurements within a Bayesian framework. As in Visual SLAM, one has usually hundreds of measurements per image, representing the (*overconstrained*) state estimation problems by a factor graph is straightforward. Furthermore, this representation can easily be extended to include other sensors measurements such as the ones of an IMU or a pressure sensor.

2.7 Manifold and Lie Groups

As stated in section 2.2, the pose of cameras are represented as elements of the $\mathbb{SE}(3)$ manifold and rotation matrices are elements of $\mathbb{SO}(3)$. These manifolds define respectively the 3D Special Euclidean group and the 3D Special Orthogonal group, which are both Lie groups. In what follows, we only define the Lie groups' properties that are of interest to use. However, the Lie theory is much deeper than what is presented here. The interested readers is referred to [206] for a concise review about the use of the Lie theory for robotics state estimation and to [10] for more detailed explanations.

A consequence of the special structures of manifolds, which do not belong to the Euclidean space, optimization of states defined as elements of such manifolds have to be performed carefully. In our case, the problem mainly comes from 3D rotation matrices $\mathbf{R} \in \mathbb{SO}(3)$, which have the following properties:

$$\mathbb{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T \cdot \mathbf{R} = \mathbf{I}_{3\times 3}, \det(\mathbf{R}) = 1 \right\} \tag{2.40}$$

As the optimization tools presented in 2.4 are only suited to states defined in Euclidean space, care has to be taken when optimizing states defined as manifolds' elements.

Smooth manifolds, such as $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$, are locally flat and there exists a corresponding Lie algebra that is defined as the tangent space at the identity element. Lie algebra is therefore defined as an Euclidean space, where any point on this space has a correspondence on the manifold space (see Fig. 2.5).

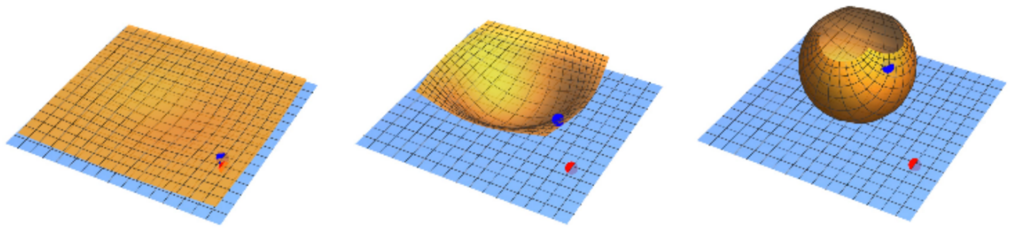


FIGURE 2.5: Illustration of the relation between $\mathbb{SO}(3)$ spheres (in orange) and their associated tangent spaces defined by $\mathfrak{so}(3)$ (in blue). The red dot and the blue dot represent correspondence between both spaces. Image taken from [108].

The mapping between the Lie algebra $\mathfrak{se}(3)$ of the $\mathbb{SE}(3)$ manifold and the Lie algebra $\mathfrak{so}(3)$ of the $\mathbb{SO}(3)$ manifold provides a way to express the current states of its Lie group elements either on the manifold space or on its local tangent space.

For $\mathbb{SO}(3)$, the mapping functions are the following:

- An **exponential map**, projecting elements from the tangent space onto the manifold:
$$\exp : \mathfrak{so}(3) \mapsto \mathbb{SO}(3) \tag{2.41}$$
- A **logarithm map**, mapping elements from the manifold to the Lie algebra tangent space:
$$\log : \mathbb{SO}(3) \mapsto \mathfrak{so}(3) \tag{2.42}$$

These exponential and logarithm mapping functions have closed-forms, that can be computed as follows.

Let $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\boldsymbol{\omega}^\wedge$ be an element of $\mathfrak{so}(3)$ associated to the rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$. Then,

$$\mathbf{R} = \exp(\boldsymbol{\omega}^\wedge) = \mathbf{I}_{3\times3} + \frac{\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|} \cdot \boldsymbol{\omega}^\wedge + \frac{1-\cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2} \cdot (\boldsymbol{\omega}^\wedge)^2 \tag{2.43}$$

where $(\cdot)^\wedge$ is here the skew-symmetric matrix operator defined as:

$$(\cdot)^\wedge : \mathbb{R}^3 \mapsto \mathfrak{so}(3) \tag{2.44}$$

If we restrict the exponential mapping to the open ball ($\|\boldsymbol{\omega}\| < \pi$), any rotation matrix $\mathbf{R}$ is uniquely defined by one $\boldsymbol{\omega}$. This way, the exponential map is bijective and its inverse is the logarithm map:

$$\log(\mathbf{R}) = \frac{\phi}{2\cdot\sin(\phi)} \cdot (\mathbf{R} - \mathbf{R}^T) \tag{2.45}$$

with:

$$\phi = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) \tag{2.46}$$

The same mapping functions exist for $\mathbb{SE}(3)$:

$$\exp : \mathfrak{se}(3) \mapsto \mathbb{SE}(3) \tag{2.47}$$

$$\log : \mathbb{SE}(3) \mapsto \mathfrak{se}(3) \tag{2.48}$$

Recalling Eq.(2.1), let $\mathbf{x} = \begin{bmatrix}\boldsymbol{\omega}^\wedge & \boldsymbol{\rho}\end{bmatrix}^T$ be an element of $\mathfrak{se}(3)$ where $\boldsymbol{\omega} \in \mathbb{R}^3$ denotes the 3 rotation coefficients, $\boldsymbol{\rho} \in \mathbb{R}^3$ determines the translation, and $\mathbf{X}$ its associated state in $\mathbb{SE}(3)$. Then,

$$\mathbf{X} = \exp(\mathbf{x}^\wedge) = \begin{bmatrix}\exp(\boldsymbol{\omega}^\wedge) & \mathbf{V}(\boldsymbol{\omega})\cdot\boldsymbol{\rho} \\ \mathbf{0}_{1\times3} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1\end{bmatrix} \tag{2.49}$$

where $(\cdot)^\wedge$ is defined as:

$$(\cdot)^\wedge : \mathbb{R}(3) \mapsto \mathfrak{so}(3) \quad ; \quad (\cdot)^\wedge : \mathbb{R}(6) \mapsto \mathfrak{se}(3) \tag{2.50}$$

The *left* Jacobian[1] of $\mathbb{SO}(3)$, $\mathbf{V}(\boldsymbol{\omega})$, has the following closed-form solution:

$$\mathbf{V}(\boldsymbol{\omega}) = \mathbf{I}_{3\times3} + \frac{1-\cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2} \cdot \boldsymbol{\omega}^{\wedge} + \frac{\|\boldsymbol{\omega}\| - \sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^3} \cdot (\boldsymbol{\omega}^{\wedge})^2 \tag{2.51}$$

The inverse operation being:

$$\log(\mathbf{X})^{\vee} = \begin{bmatrix} \log(\mathbf{R})^{\vee} \\ \mathbf{V}(\boldsymbol{\omega})^{-1} \cdot \mathbf{t} \end{bmatrix} \tag{2.52}$$

with $\log(\mathbf{R})$ the logarithm map of $\mathbb{SO}(3)$ defined in Eq.(2.45) and:

$$\mathbf{V}(\boldsymbol{\omega})^{-1} = \mathbf{I}_{3\times3} - \frac{1}{2} \cdot \boldsymbol{\omega}^{\wedge} + \frac{1 - \frac{\|\boldsymbol{\omega}\| \cdot \cos(\|\boldsymbol{\omega}\|/2)}{2 \cdot \sin(\|\boldsymbol{\omega}\|/2)}}{\|\boldsymbol{\omega}\|^2} \cdot (\boldsymbol{\omega}^{\wedge})^2 \tag{2.53}$$

For notational convenience, we further define the following operators that will be used in the next chapters:

$$\mathrm{Exp} : \mathbb{R}^6 \mapsto \mathbb{SE}(3) \quad ; \quad \mathrm{Log} : \mathbb{SE}(3) \mapsto \mathbb{R}^6 \tag{2.54}$$

$$\mathrm{Exp} : \mathbb{R}^3 \mapsto \mathbb{SO}(3) \quad ; \quad \mathrm{Log} : \mathbb{SO}(3) \mapsto \mathbb{R}^3 \tag{2.55}$$

$$\mathrm{Exp}(\cdot) = \exp(\cdot^{\wedge}) \quad ; \quad \mathrm{Log}(\cdot) = \log(\cdot)^{\vee} \tag{2.56}$$

These mapping functions can finally be used to perform nonlinear least-squares optimization on the manifolds' tangent spaces while keeping the manifold structure of the states. The optimization operations that are going to be performed throughout this thesis will highly rely on the Lie groups-Lie algebra correspondence defined in this section.

2.8 Conclusion

In this chapter, we have reviewed most of the mathematical tools that we will use in the next chapters. The application of these different concepts will

[1]Left and right Jacobians of $\mathbb{SO}(3)$ denotes the Jacobians related to a perturbation applied on the left part or on the right part of a rotation matrix [206].

be detailed throughout the thesis to perform the tasks of visual localization, mapping and sensors fusion.

Chapter 3

Underwater Features Tracking

Contents

3.1 Introduction

The problem of features tracking is one of the cornerstones of visual localization. Indeed, Visual SLAM (VSLAM) is built on the fact that, by observing real-world 3D points in successive 2D images (given a small motion between each), one is able to estimate the pose (position and orientation) of the camera with respect to these 3D observations. The first key for accurate localization is therefore to find the projection of these 3D points into the captured images. Furthermore, if one is able to determine the image locations of an unknown 3D point between two images taken from two different points of view, these two observations can be leveraged to estimate the position of this 3D point. As camera's pose estimation from known 3D observations and 3D mapping are the building blocks of any VSLAM system, very accurate features tracking is required. This features tracking problem is a special case of what is referred to as *data association* in the robotics community.

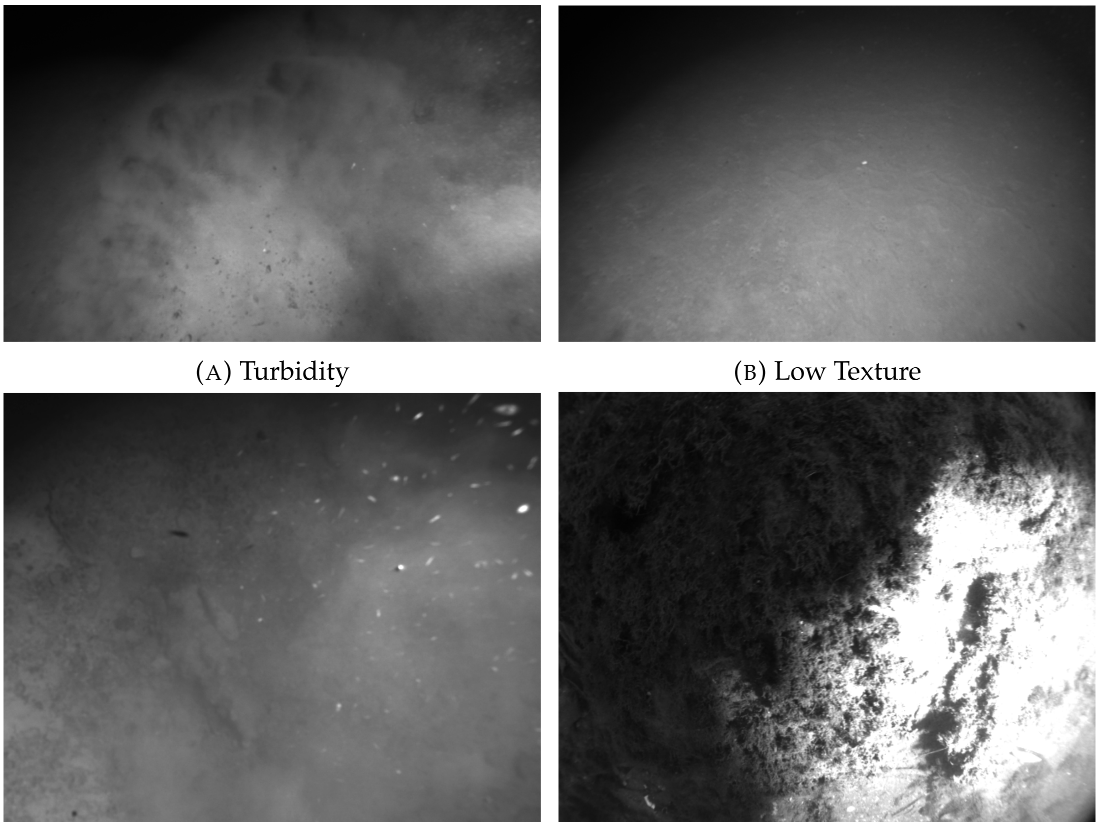
In this chapter, we conduct a thorough evaluation of several features tracking strategies in the underwater context. We mainly investigate and compare the performance of *indirect* and *direct* tracking methods on different underwater sets of images, each exhibiting some of the classical underwater visual challenges. First, we formally define the problem of features tracking and detail the workflow of *indirect* and *direct* tracking paradigms. Then, we review some related works and compare them to our approach. Next, we present the sets of images used to perform our analysis. The evaluation is finally presented by, first, defining a protocol and some evaluation criteria, and, second, analyzing the obtained results.

3.2 Problem Statement

The classical approach for features tracking consists in detecting salient points in an image (refered to as features or keypoints, interchangeably) and trying to find their new locations in the next images. The whole challenge here is hence to find appropriate ways of determining these new locations from visual data only.

In underwater environment, the tracking of features is disturbed by the poor imaging conditions, mainly due to the presence of turbidity and backscattering. Moreover, the tracking is even more challenging because of artificial illumination and the presence of many texture-less areas. Illustrations of these visual challenges are displayed in Fig. 3.1 The suitability of a tracking method in this context hence depends on its robustness to these visual disturbances.

Features tracking is mainly solved by either indirect or direct methods. Indirect methods rely on an encoding of the appearance of detected features through the use of descriptors. The problem of features tracking is then



(A) Turbidity (B) Low Texture

(C) Backscattering (D) Strong illumination variations

FIGURE 3.1: Typical underwater optical perturbations.

solved by computing distances between different descriptors to find correspondences. On the other hand, direct methods directly use the pixel values around detected features to track them in other images.

We next review in more detail these two different tracking strategies.

3.3 Tracking Strategies

The tracking problem consists in finding corresponding pixels between two images $\mathbf{I}_1$ and $\mathbf{I}_2$. In VO and VSLAM, the tracking is often performed by searching for the corresponding pixels in $\mathbf{I}_2$ from a set of keypoints detected in $\mathbf{I}_1$ (Fig.3.2).

We now review two different strategies to solve the features tracking problem: indirect tracking and direct tracking.

3.3.1 Indirect Tracking

Indirect tracking refers to the use of descriptors as a tracking tool. When extracting features from an image, each detected keypoint is described and

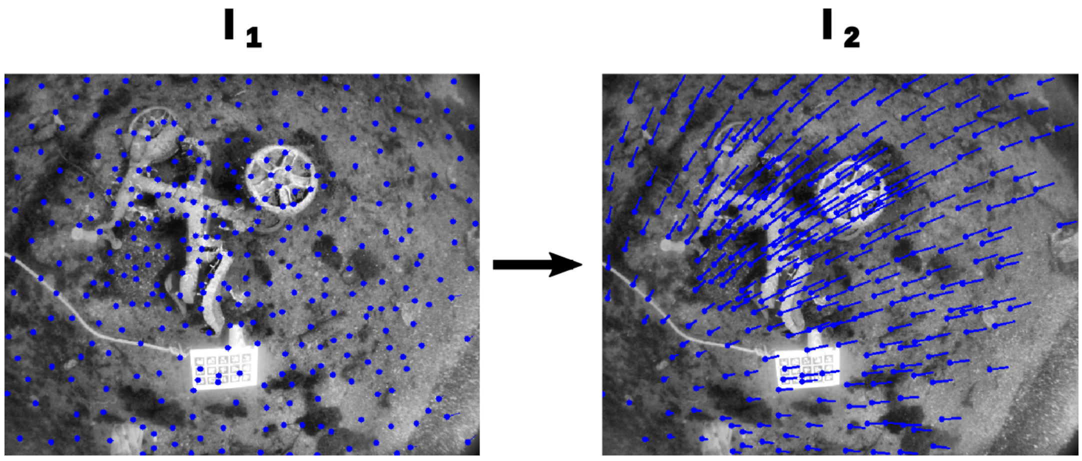


FIGURE 3.2: Illustration of features tracking between two images $\mathbf{I}_1$ and $\mathbf{I}_2$. Detected keypoints in $\mathbf{I}_1$ (blue dots) are tracked into $\mathbf{I}_2$. The blue dots in $\mathbf{I}_2$ represent the new keypoints position and the blue segment represent the displacement of the keypoints between $\mathbf{I}_1$ and $\mathbf{I}_2$.

associated to an encoding vector characterizing its appearance from the surrounding pixels (Fig.3.3). This method models the visual appearance of keypoints (as well as some geometrical properties such as scale or orientation for invariance) into moderately high dimensional vectors (typically 128 to 1024). These vectors are called descriptors and can be compared efficiently. This comparison is done by computing the norm of their differences, referred to as the matching distance, to obtain scalar-valued similarity measurements between them.

More formally, the indirect tracking problem can be modeled as follows. Given $\mathbf{\Omega}$, a square image patch:

$$\mathbf{\Omega}(u,v) \ \setminus \ \mathbf{\Omega} \in \mathbb{N}^{n \times n}, u \in \mathbb{N}, v \in \mathbb{N}, n \in \mathbb{N} \tag{3.1}$$

with (u,v) the pixel coordinates of the patch center and $n \times n$ the size of the patch. The descriptor function f can be defined as :

$$f : \mathbb{N}^{n \times n} \mapsto \mathbb{R}^k \ \ , f(\mathbf{\Omega}(\mathbf{x})) = \mathbf{d} \ \ , k \in \mathbb{N} \tag{3.2}$$

where k is the dimension of the descriptor and $\mathbf{d}$ is the descriptor of the keypoint $\mathbf{x}$ with coordinates (u, v).

We define the function g that calculates a distance between two descriptors:

$$g(\mathbf{d}_i, \mathbf{d}_j) = \mathbf{norm}\left(\mathbf{d}_j - \mathbf{d}_i\right) s \ , \ s \in \mathbb{R} \tag{3.3}$$

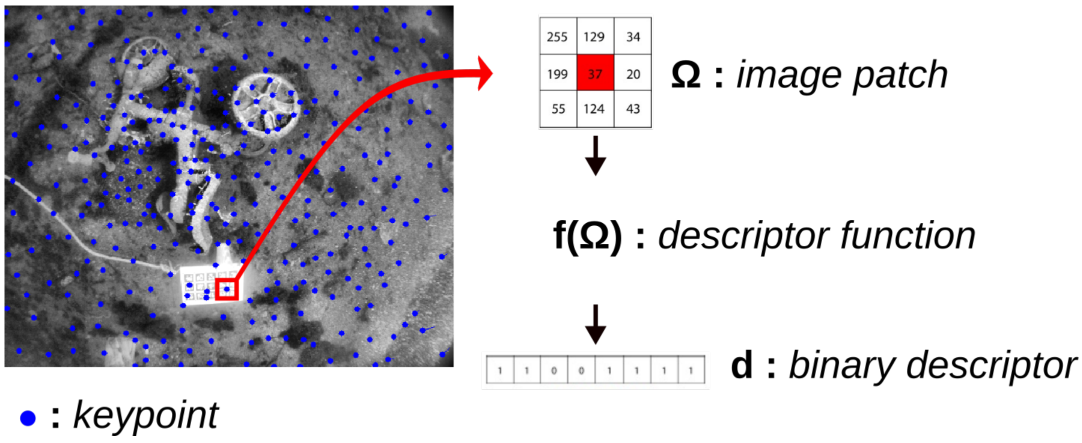


FIGURE 3.3: Illustration of the features description process. A square image patch $\mathbf{\Omega}$ is extracted around each detected keypoint (blue dots). A description function $\mathbf{f}$ is then applied on this patch in order to compute a vector $\mathbf{d}$ that describes the keypoints (with a binary descriptor here). A similarity score between descriptors can then be computed in order to find corresponding features between two images.

Then, the problem of features tracking is the following. Given $\mathfrak{K}_1$ and $\mathfrak{K}_2$, the sets of extracted features in images $\mathbf{I}_1$ and $\mathbf{I}_2$, the tracking consists in looking for the correct matching pair ($\mathbf{x}_i$,$\mathbf{x}_j$), where $\mathbf{x}_i \in \mathfrak{K}_1$ and $\mathbf{x}_j \in \mathfrak{K}_2$ are the image projections of a same 3D point, for each feature. The computation of the distance between the descriptors provides then a way of correctly matching the features.

We find two families of descriptors, real-valued and binary. Real-valued descriptors are usually the most accurate, as they can encode more information than binary ones, but they come at a significantly higher computational cost (see Table 3.1 for timing comparison). On the opposite, binary descriptors are quite fast to compute as they are merely the result of pixels' intensities comparison performed following a predefined pattern (*e.g.* (0 if $\mathbf{I}(u,v) \leq \mathbf{I}(u+i,v+j)$ and 1 otherwise).

TABLE 3.1: Average run-time per keypoint for keypoint detection and description. Extracted from [159].

	Methods						
	Binary					*Real-Valued*	
	ORB	BRISK	FAST	BRIEF	FREAK	SURF	SIFT
Detection time (μs)	11.8	5.34	0.39	x	x	60.5	217.8
Description time (μs)	4.2	10.6	x	3.8	6.15[1]	117.1	448.6

[1]Value extracted from [3] and adjusted to match the evaluation performed in [159]

The computation of the matching distance is also less complex with binary descriptors compared to real-valued ones, as it relies on the Hamming norm between bits – efficiently computed with a XOR operation – instead of classical ℓ_2 norm.

In the following, we use the SIFT [146] and SURF [15] real-valued descriptors and the ORB [191], BRISK [136], BRIEF [26] and FREAK [3] binary ones. Details about these methods and how they differ from each others can be found in [14, 158].

The most basic way of solving the matching problem is to proceed with an exhaustive solution by searching for the nearest neighbor of each descriptor. The descriptor distance between all the features extracted in the images $\mathbf{I}_1$ and $\mathbf{I}_2$ is calculated. Then, the most likely matchings can be recovered by looking for the pairs ($\mathbf{x}_i$,$\mathbf{x}_j$) with the lowest descriptor distance. In order to limit the number of wrong matchings, the calculation of the descriptor distances can be performed twice. Once from the features in $\mathbf{I}_1$ towards the features in $\mathbf{I}_2$ and once in the inverse way. As it is very unlikely that there is a perfect one-to-one matching between the features in both images, different results might be obtained between both steps. The most likely matchings are then the pairs with the lowest distance in both cases ($\mathbf{I}_1 \rightarrow \mathbf{I}_2$ and $\mathbf{I}_2 \leftarrow \mathbf{I}_1$). An additional way of making more robust the matching process is to look at the second lowest distance for each selected pair and to only validate it if the difference between the lowest matching distance and the second lowest is higher than a threshold (typically set as 80 % of the second lowest distance [146]). This way, ambiguous matchings can be removed and only distinctive ones are kept. However, brute-force tracking can be computationally quite expensive. In the context of VSLAM and VO, the matching process can be improved and alleviated by using a motion prior in order to limit the search area. In this case, for each feature $\mathbf{x}_i$ in $\mathbf{I}_1$, the matching distance is only computed between the features of $\mathbf{I}_2$ located within a window around the predicted location of $\mathbf{x}_i$ in $\mathbf{I}_2$. Such improvements both increase the robustness and the computational speed of the matching process.

3.3.2 Direct Tracking

Direct tracking methods define the tracking problem as the problem of finding the correct pairs ($\mathbf{x}_i$,$\mathbf{x}_j$) by working directly on the pixel's intensities. In the direct tracking formulation, one only needs to detect features in the first image $\mathbf{I}_1$. The location of each feature in the second image is then generally estimated by computing the intensity difference between a patch around $\mathbf{x}_i$ in $\mathbf{I}_1$ and patches in $\mathbf{I}_2$. The new location of $\mathbf{x}_i$ in $\mathbf{I}_2$ is then estimated as the center of the patch with the minimum intensity difference in $\mathbf{I}_2$. As in the indirect tracking, the tracking problem can either be solved in a brute-force way, by sliding a patch around every possible locations in $\mathbf{I}_2$, or using a motion prior to limit the search area.

In order to estimate similarity between two intensity patches several metrics can be used. The Sum of Squared Differences (SSD) and Zero-Mean Normalized Cross-Correlation (ZNCC) are two examples of direct tracking:

$$\mathbf{SSD}(\mathbf{\Omega}_1, \mathbf{\Omega}_2) = \sum_i^N \sum_j^N (\mathbf{\Omega}_1(i,j) - \mathbf{\Omega}_2(i,j))^2 \tag{3.4}$$

$$\mathbf{ZNCC}(\mathbf{\Omega}_1, \mathbf{\Omega}_2) = \frac{\frac{1}{N^2} \sum_i^N \sum_j^N [(\mathbf{\Omega}_1(i,j) - \mu_1) \cdot (\mathbf{\Omega}_2(i,j) - \mu_2)]}{\sigma_1 \cdot \sigma_2} \tag{3.5}$$

where $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ are image patches of size $N \times N$ pixels, (μ_1, σ_1) and (μ_2, σ_2) are, respectively, the mean and standard deviations of the intensities of the patches $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$.

In the context of sequential features tracking, an elegant way to perform a direct tracking is to use the notion of optical flow. Optical flow defines the velocity of each pixel in a stream of images. Dense optical flow methods consist in estimating the motion between two images trough the computation of a 2D velocity vector field. The same idea can be applied for sparse tracking problems such as ours. The Lucas-Kanade [9] algorithm is a solution to sparse optical flow estimation that can be performed iteratively [21].

The iterative form of the Lucas-Kanade method works as follows.

Given two images $\mathbf{I}_t$ and $\mathbf{I}_{t+1}$, we want to find in $\mathbf{I}_{t+1}$ the new position of a keypoint $\mathbf{x}(t)$, with known position in $\mathbf{I}t$. If one makes the following assumptions:

- the brightness stayed constant between t and $t+1$
- the displacement between $\mathbf{I}_{t+1}$ and $\mathbf{I}_{t+1}$ is small

Then, we can express the new position of $\mathbf{x}$ as: $\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{dx}$, with

$$\mathbf{x}(t) = \begin{bmatrix} u \\ v \end{bmatrix} \text{ , } \mathbf{x}(t+1) = \begin{bmatrix} u + du \\ v + dv \end{bmatrix} \text{ , } \mathbf{dx} = \begin{bmatrix} du \\ dv \end{bmatrix} \tag{3.6}$$

Moreover, if ones assumes an identical motion for neighboring pixels, we can extend the previous statement over image patches of size $N \times N$:

$$\sum_i^N \sum_j^N \Big(\mathbf{I}_{t+1}(u + du, v + dv) - \mathbf{I}_t(u, v) \Big) = 0 \tag{3.7}$$

The tracking problem is hence solved by finding the 2D vector **dx** which satisfies the previous equation. The problem being overconstrained and the inputs of the problem being non-linear, we can solve it by a non-linear least-squares method. We define a small perturbations around (du, dv) as $(du + \delta u, dv + \delta v)$.

$$\sum_i^N \sum_j^N \Big(\mathbf{I}_{t+1}(u_i + du + \delta u, v_j + dv + \delta v) - \mathbf{I}_t(u_i, v_j) \Big)^2 = 0 \tag{3.8}$$

Then, applying a Taylor first-order propagation, one gets:

$$\sum_i^N \sum_j^N \left(\mathbf{I}_{t+1}(u_i + du, v_j + dv) - \mathbf{I}_t(u_i, v_j) + \frac{\partial \mathbf{I}_{t+1}}{\partial u} \cdot \delta u + \frac{\partial \mathbf{I}_{t+1}}{\partial v} \cdot \delta v \right)^2 = 0 \tag{3.9}$$

In this context, the resulting Jacobian appends to be the horizontal and vertical gradients of the image:

$$\nabla \mathbf{I} = \begin{bmatrix} \frac{\partial \mathbf{I}_{t+1}}{\partial u} & \frac{\partial \mathbf{I}_{t+1}}{\partial v} \end{bmatrix} \tag{3.10}$$

leading to:

$$\sum_i^N \sum_j^N \left(\mathbf{I}_{t+1}(u_i + du, v_j + dv) - \mathbf{I}_t(u_i, v_j) + \nabla \mathbf{I} \cdot \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} \right)^2 = 0 \tag{3.11}$$

The minimum of this cost function is finally found by solving the normal equations:

$$\begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = - \sum_i^N \sum_j^N \Big(\nabla \mathbf{I}^T \cdot \nabla \mathbf{I} \Big)^{-1} \cdot \nabla \mathbf{I}^T \cdot \delta \mathbf{I} \tag{3.12}$$

$$\delta \mathbf{I} = \mathbf{I}_{t+1}(u_i + du, v_j + dv) - \mathbf{I}_t(u_i, v_j) \tag{3.13}$$

$$\nabla \mathbf{I}^T \cdot \nabla \mathbf{I} = \begin{bmatrix} \nabla \mathbf{I}_{uu} & \nabla \mathbf{I}_{uv} \\ \nabla \mathbf{I}_{vu} & \nabla \mathbf{I}_{vv} \end{bmatrix} \tag{3.14}$$

where $\delta\mathbf{I}$ is the current intensities residual and $\nabla\mathbf{I}^T \cdot \nabla\mathbf{I}$ is the second derivative of the function, which can be estimated from the image gradients.

This problem is solved iteratively by updating the 2D vector after each iteration:

$$du \leftarrow du + \delta u \quad ; \quad dv \leftarrow dv + \delta v \tag{3.15}$$

This optimization problem can be performed at multiple resolutions by building pyramids of the images $\mathbf{I}_{t+1}$ and $\mathbf{I}_t$. Starting at a coarse resolution and refining the solution layer by layer in the pyramids, the formulation of this problem can handle larger pixels displacements. Last, the $\delta\mathbf{I}$ residual can be used to validate or not a tracking (by setting a minimum residual to reach for instance).

In this thesis, we use such a pyramidal implementation of the Lucas-Kanade algorithm. The details of the exact algorithm are given in [21].

We couple this tracking algorithm with the detection of Harris corners computed through the method of Shi and Tomasi [215, 199]. Such a combination is often referred to as the KLT (Kanade-Lucas-Tomasi) tracker in the literature.

3.4 Related Works

In the underwater context, [7] evaluates the use of SURF features [15] in their SLAM algorithms and assess improvements in the localization but their method does not run in real-time. The authors of [200] evaluates several combinations of feature detectors and descriptors and they show that keypoints detected the SURF or Shi-Tomasi detector [199] matched by ZNCC (see Eq.(3.5)) performed better for visual state estimation than the other considered methods. However, these works did not include the evaluation of binary features such as BRIEF [26] or ORB [191] nor of optical-flow, which are widely used methods for features tracking in VO and VSLAM. In [100], binary descriptors have been evaluated along real-valued ones in the context of VSLAM for a terrestrial wheeled robot. The results obtained are in favor of SIFT but they also show that BRIEF perfoms better than other binary descriptors. In [87, 45], many feature detectors are evaluated on underwater images with a focus on their robustness to turbidity. They follow the evaluation protocol of [158] by using the detectors repeatability as their metric. Robustness to turbidity is essential for underwater visual localization, but only evaluating repeatability of the detectors does not ensure good features tracking capacity. Indeed, ambiguities can still arise when trying to match these features between two images.

Our features tracking evaluation differs from these previous works in that we analyze the features tracking efficiency of a wide range of feature detectors and tracking methods in the specific context of underwater VO and VSLAM.

3.5 Underwater Sets of Images

We have used two different sets of images to conduct the evaluation of the different tracking methods. The first one is the TURBID dataset [45], which consists of three series of static pictures of a printed seabed taken in a pool (Fig. 3.4). Turbidity was simulated on these images by adding in the water a controlled quantity of milk between two shots. The second one consists of a sequence of images extracted from a video recorded by a moving camera close to the seabed (Fig. 3.5). This sequence exhibits the typical characteristics of underwater images: low texture and repetitive patterns. As this set is a moving one, we will refer to it as the VO set. On both sets, all the images used are resized to 640 $\times$ 480 and gray-scaled to fit the input format of classical VO methods.

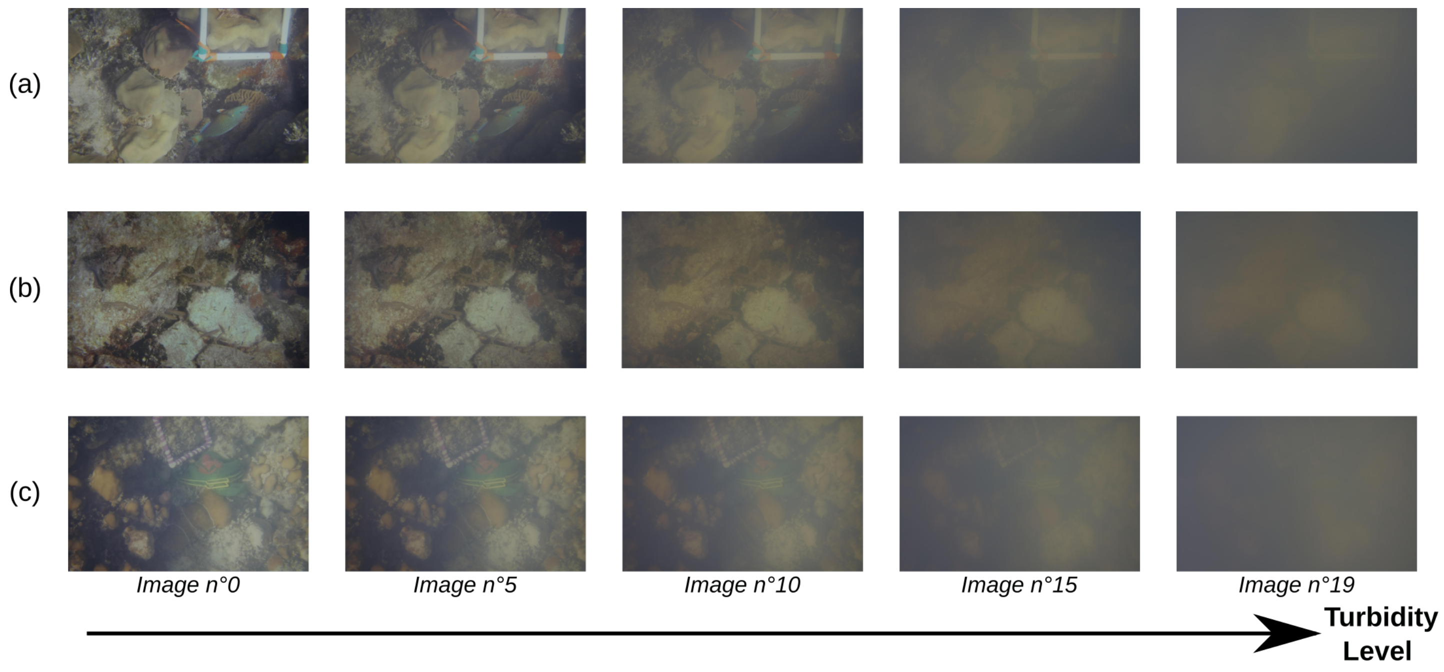


FIGURE 3.4: Images from the TURBID dataset [45].

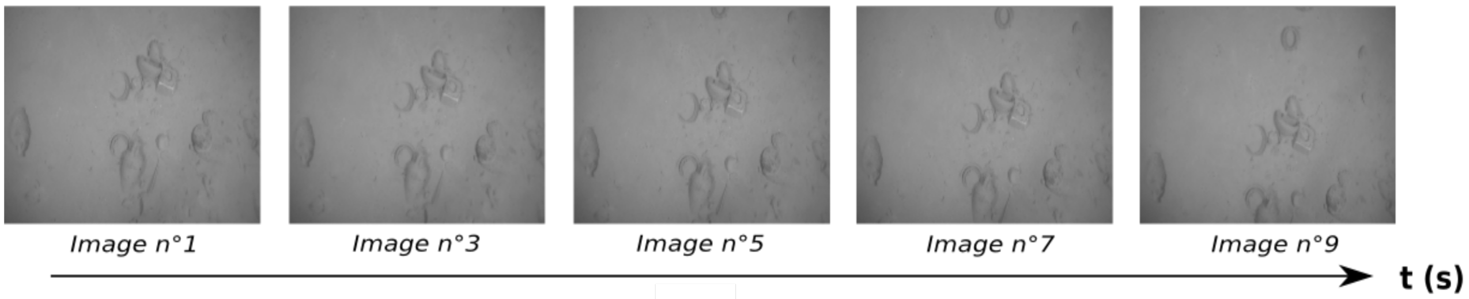


FIGURE 3.5: Images from the VO set acquired on a deep antic shipwreck (depth: 500 meters, Corsica, France) - Credit: DRASSM (French Department of Underwater Archaeological Research).

3.6 Evaluation Metrics

In order to evaluate and analyze the performance of the different features tracking strategies, we need to define some evaluation metrics. The choice of these metrics is based on our expectations in terms of tracking quality and accuracy in a visual localization context. Ideally, we would like to use a method able to track every observable keypoints from one image to another, with perfect keypoints' localization and no mis-matching (that is matching wrong pairs of keypoints).

First, we use the Turbid sets of static images, degraded by a simulated increase of turbidity. As the images are static, we have a ground-truth to evaluate the tracking quality and accuracy here.

We define the tracking quality, for indirect tracking methods, as the distinctiveness of the features in comparison with their nearest neighbors. In addition, we define the tracking accuracy as the pixel error between the tracked features' positions and the real one. This second criterion is both used for direct and indirect tracking methods. These evaluations allow us to measure the robustness to turbidity of the different methods.

Then, we use the both the Turbid sets and the VO set of images to evaluate the efficiency of the tracking methods on a real-case scenario. We define the tracking efficiency as being proportional to the number of successfully tracked features.

We now give in-depth details about how we quantify these criteria and how we can interpret them.

3.6.1 Indirect tracking evaluation criteria

For the indirect methods, we want to evaluate the distinctiveness of the extracted features in small areas. Indeed, as presented in section 3.3.1, the matching selection will be based on the similarity scores between descriptors located in a bounded area. Intuitively, the more distinctive the features are in a small area, the better the matching quality is, as it does not suffer from matching ambiguities (several candidates with high similarity). To do so, we run an evaluation by taking successive image pairs, extracting features in both images and matching each feature of the first image to the features of the second image, located in a 20×20 pixels area around it.

We define $\mathfrak{D}_i$ as the set of features extracted in the first image of each pair and $\mathfrak{D}_{ij}$ as the sets of features from the second image located within the 20×20 pixels window around an element i of $\mathfrak{D}_i$. The criteria that we use to evaluate the performance are the following statistics [2]:

1. The statistics over the matching distance: the matching distance with all features located in a close area

$$\mu_1 = \frac{1}{N}\sum_i^N \left(\frac{1}{M}\sum_j^M (g(\mathbf{d}_i - \mathbf{d}_j)) \right) \tag{3.16}$$

$$\sigma_1 = \frac{1}{N}\sum_i^N \sqrt{\left(\frac{1}{M-1}\sum_j^M (g(\mathbf{d}_i - \mathbf{d}_j) - \mu_1)^2 \right)} \tag{3.17}$$

with $\mathbf{d}_i \in \mathfrak{D}_i$, $\mathbf{d}_j \in \mathfrak{D}_{ij}$ and $g(\cdot)$ taken from Eq.(3.3) and N and M the number of features in the $\mathfrak{D}_i$ and $\mathfrak{D}_{ij}$ sets.

2. The statistics over the minimum matching distance: the minimum matching distance found for each feature

$$\mu_2 = \frac{1}{N}\sum_i^N \left(g\left(\mathbf{d}_i - \mathbf{d}_{min_j}\right) \right) \tag{3.18}$$

$$\sigma_2 = \sqrt{\left(\frac{1}{N-1}\sum_i^N \left(g\left(\mathbf{d}_i - \mathbf{d}_{min_j}\right) - \mu_2 \right)^2 \right)} \tag{3.19}$$

with $\mathbf{d}_{min_j}$ the most similar descriptor to $\mathbf{d}_i$ in $\mathfrak{D}_{ij}$.

[2] By statistics we understand the mean and standard deviation here.

3. 80% of the 2nd minimum matching distance: the 2nd minimum matching distance multiplied by 0.8 for each feature

$$\mu_3 = \frac{1}{N}\sum_i^N \left(0.8 \cdot g\left(\mathbf{d}_i - \mathbf{d}_{2^{nd}min_j}\right)\right) \tag{3.20}$$

$$\sigma_3 = \sqrt{\left(\frac{1}{N-1}\sum_i^N \left(0.8 \cdot g\left(\mathbf{d}_i - \mathbf{d}_{2^{nd}min_j}\right) - \mu_3\right)^2\right)} \tag{3.21}$$

with $\mathbf{d}_{2^{nd}min_j}$ the second most similar descriptor to $\mathbf{d}_i$ in $\mathfrak{D}_{ij}$.

4. The good matching distance: the distance relating two matched features considered as inliers (assessed by an error less than 2 pixels)

$$\mu_4 = \frac{1}{N}\sum_i^N \left(g\left(\mathbf{d}_i - \mathbf{d}_{good_j}\right)\right) \tag{3.22}$$

$$\sigma_4 = \sqrt{\left(\frac{1}{N-1}\sum_i^N \left(g\left(\mathbf{d}_i - \mathbf{d}_{good_j}\right) - \mu_4\right)^2\right)} \tag{3.23}$$

with $\mathbf{d}_{good_j}$ the most similar descriptor to $\mathbf{d}_i$ in $\mathfrak{D}_{ij}$ given an error of less than 2 pixels between both related features.

5. The statistics over the tracking error: the pixel error between matched features (*i.e.* the ones with the most similar descriptors)

$$\mu_5 = \frac{1}{N}\sum_i^N \left\|\mathbf{x}_{\mathbf{d}_i} - \mathbf{x}_{\mathbf{d}_{min_j}}\right\|_2 \tag{3.24}$$

$$\sigma_5 = \sqrt{\left(\frac{1}{N-1}\sum_i^N \left(\left\|\mathbf{x}_{\mathbf{d}_i} - \mathbf{x}_{\mathbf{d}_{min_j}}\right\|_2 - \mu_5\right)^2\right)} \tag{3.25}$$

with $\mathbf{x}_{\mathbf{d}_i}$ and $x_{\mathbf{d}_{min_j}}$ the pixel coordinates associated to the descriptors $\mathbf{d}_i$ and $\mathbf{d}_j$ and the pixel error represented by the euclidean norm.

The first four criteria relate to a prediction of the tracking quality of the methods. Indeed, if all these distances are close, one cannot expect good results in a practical VSLAM scenario, as many ambiguities will arise in the matching process. On the opposite, if the minimum matching distances can be strongly distinguished from the others, numerous correct matching are expected to be found. The last criterion gives an insight of the tracking accuracy that can be expected in similar imaging conditions.

3.6.2 Direct tracking evaluation criterion

For the direct methods, we only use the tracking error metric. The evaluation is again run by taking successive image pairs.

As these methods look for the minimum photometric residual between patches extracted from two images, we evaluate the expected tracking accuracy as the pixel errors between the found minima and the groundtruth. Here, we evaluate the results obtained with the SSD method and with the KLT. For the SSD method, we first extract keypoints in the first image, take a patch of 15×15 pixels around each keypoint and finally compute the SSD, using Eq.(3.4), between this patch and patches in the second image by moving in a 20×20 pixels window around the original keypoint. The pixel error between the found minimum and the original keypoint location will be averaged over all extracted keypoints to obtain an estimate of the tracking accuracy. For the KLT method the tracking is performed using Eq.(3.11) with a patch of 15×15 pixels and three layers.

3.6.3 Tracking efficiency evaluation

In a second time, we evaluate the tracking efficiency of each method on the Turbid dataset. To do so, we divide each image in 500 cells and try to extract one feature per cell (this technique is sometimes referred to as bucketing in the VO literature [195]). Then, we try to track each feature in the following image. We consider a tracking successfull if the pixel error is less than 2 pixels. For the indirect methods, the search space is limited to a 40×40 pixels window around the original feature in order to simulate the matching case of VO scenarios. The best candidate among the found features is simply selected as the one with the minimum matching distance. Within the direct methods here we only use the KLT. For the KLT method, we recover, if any, the location of the found minimum. Note that, the KLT method acts locally and starts its search at the exact location of the original feature. Hence, to avoid perfect initialization of the search space for the KLT, we apply a virtual translation of 10 pixels to the target image. By doing so, we avoid advantaging any method in front of another.

The last analysis that we perform is on the VO set of images. On this set, we apply a pure VO scenario to our analysis: we try to track features extracted in a reference image (a *keyframe*) in all the following images. This scenario is typical when one is tracking a keypoint over several images, waiting for enough parallax between the original keypoint location and the current one in order to triangulate it accurately. Here, we are again interested in the tracking efficiency of each method. However, as we don't have any ground-truth to refer to on this set, we apply some classical outlier removal techniques for features tracking in a VO context. Once the tracking of features in each new image is completed, we check the epipolar consistency between the tracked features by computing the Essential Matrix in a RANSAC scheme and remove the detected outliers (see section 2.3.4 page 27). Furthermore, for the KLT, a forward-backward check is applied to remove any ambiguous track. This forward-backward check is simply done by running the optical flow algorithm twice, once from the current image toward the target image and once from the target image toward the current image (using the tracked feature's location as the new starting point). If there is a difference higher than 1 pixel between the original feature and the backward one, the feature is removed. Otherwise, the protocol is the same as the previous one. The only deviations are that, for the indirect methods, the matching is always performed between the first image and the following ones and, for the KLT method, features are only extracted in the first image and then tracked image by image (then if a feature is lost at some point, it cannot be recovered later as with indirect methods). Furthermore, the center of the search space window for the indirect methods is set by the average disparity between the keyframe and the features successfully tracked in the previous image. As the VO set contains images of the seabed, the assumption that every pixels are approximately moving in the same direction and with the same magnitude is true here.

For these last two evalutations, another consideration is the number of successfully detected features in each image. Indeed, if a given method is not able to detect many features in the images it will most likely perform poorly in a VO application. Hence, we also record the number of successfully detected features per image.

Summary

In short, the criteria that we are interested in for evaluating the different tracking methods are:

- The tracking quality, defined as the evaluation of features distinctiveness compared with their close neighbors (indirect methods only)
- The tracking error, defined as the pixel error between the tracked features and the groundtruth
- The tracking efficiency, defined by the number of successfully tracked features on a sequence of images

3.7 Evaluation Results

We now compare the results obtained for the different methods. We have used the OpenCV library implementation of all the tested methods but the SSD, which we implemented ourselves.

3.7.1 Tracking Quality and Tracking Error Analysis

3.7.1.1 Evaluation Criteria Recall

We start by recalling what we expect to analyze from the tracking quality statistics of the different indirect methods. As these kinds of methods rely on descriptors as a mean of measuring the similarity between different features, an efficient tracking method should present high similarity between related features and low similarity between unrelated ones. As the similarity score is computed as the norm of the descriptors difference, the higher the similarity is, the lower the norm is. The norm is referred to as the matching distance. Here, we have computed the matching distances between features using a search window. So, what we analyze is more specifically the similarity differences between features located in a precise area.

Hence, from an efficient indirect method, we first expect the matching distance of two related features to be unambiguously lower than the average of the matching distances obtained with all the others features in the searching area. Otherwise, it would mean that the correct matching is almost impossible to recover. Secondly, we also expect the correct matching distance to be significantly lower than the second lowest distance. As explained in section 3.3.1, a classical ambiguity checking is to ensure that the minimum distance is lower than 80% of the second one to validate a matching. By adding to

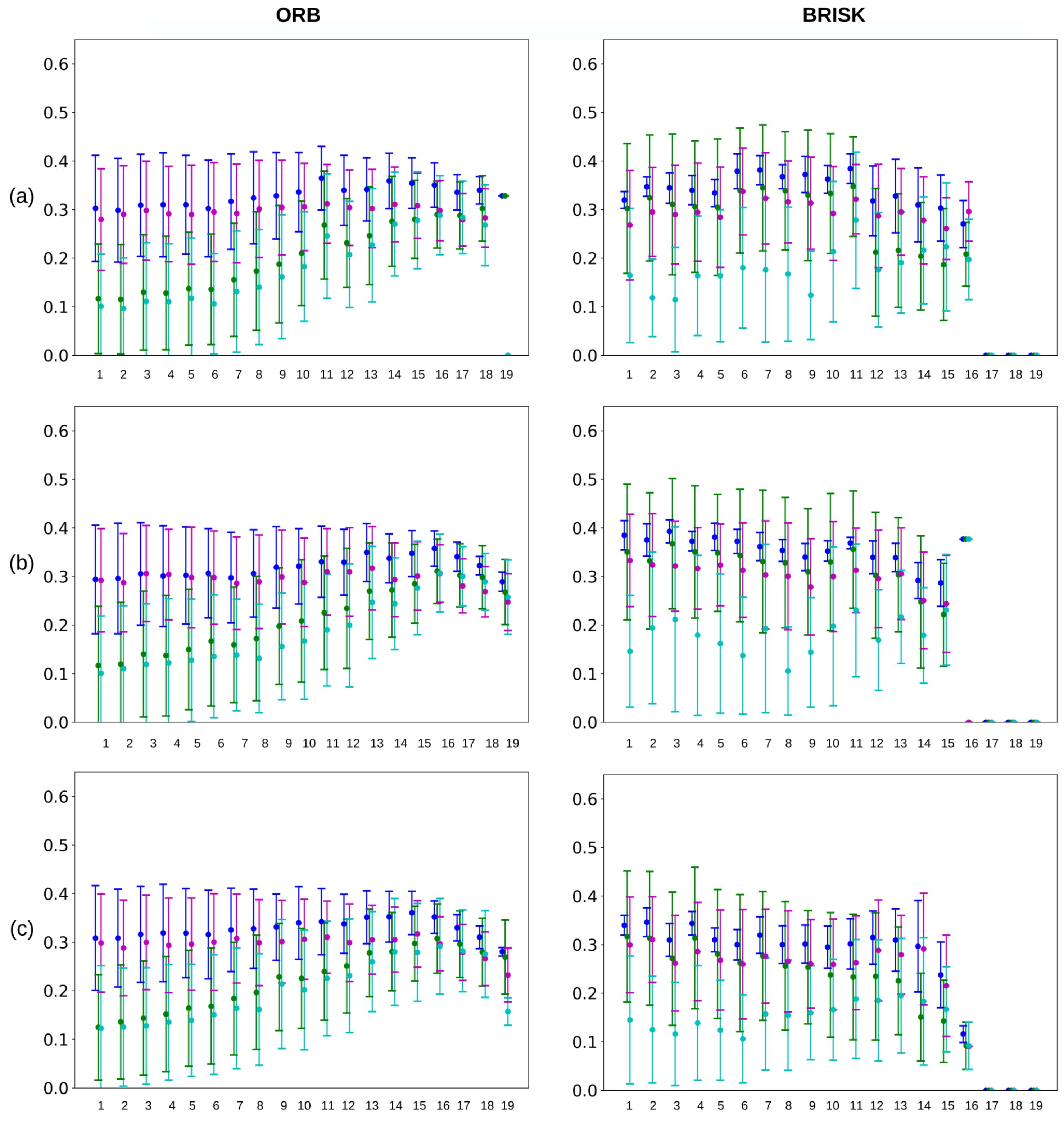


FIGURE 3.6: Tracking quality statistics for ORB and BRISK. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the normalized matching distance.

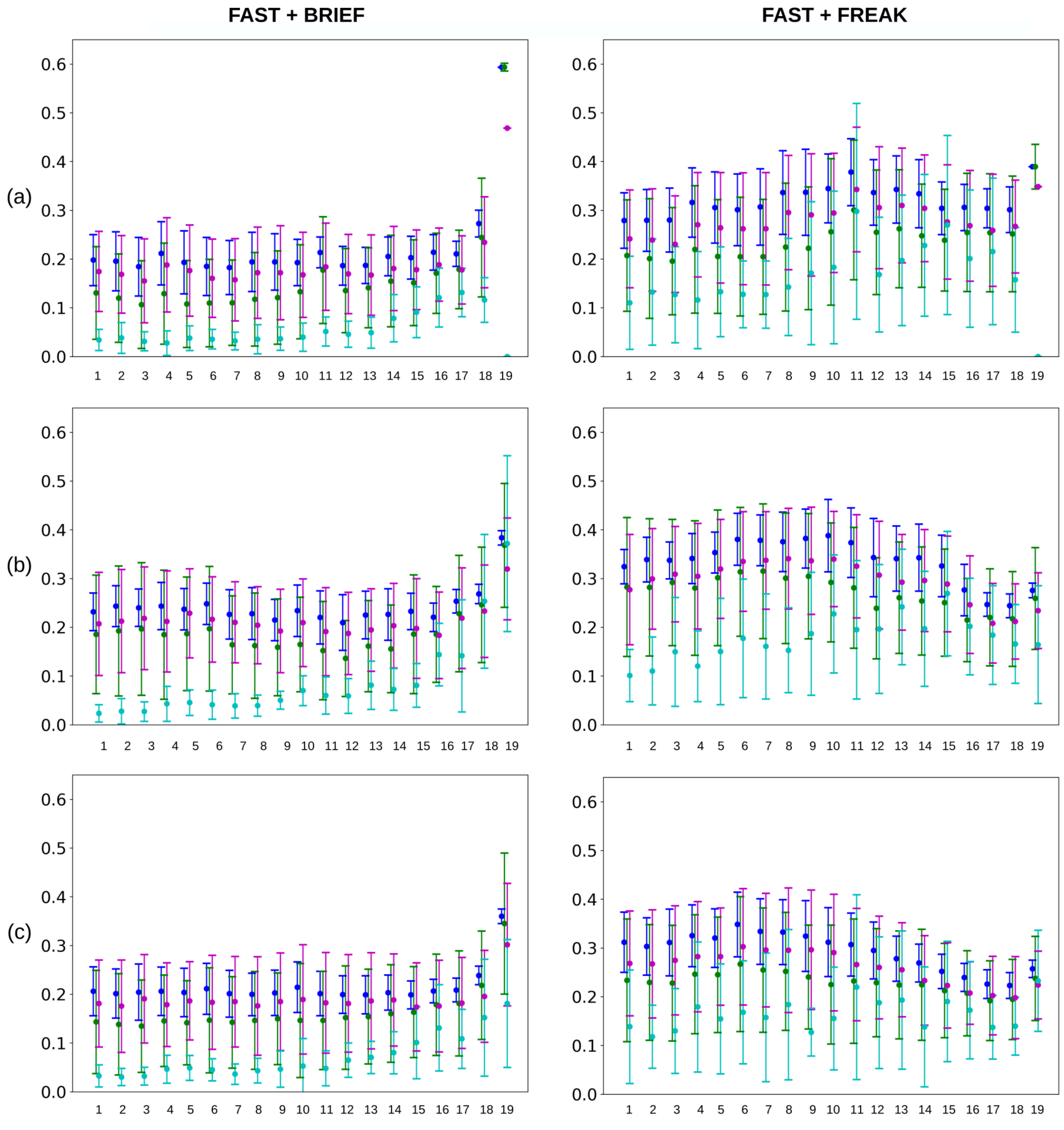


FIGURE 3.7: Tracking quality statistics for FAST combined with BRIEF and FREAK. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the normalized matching distance.

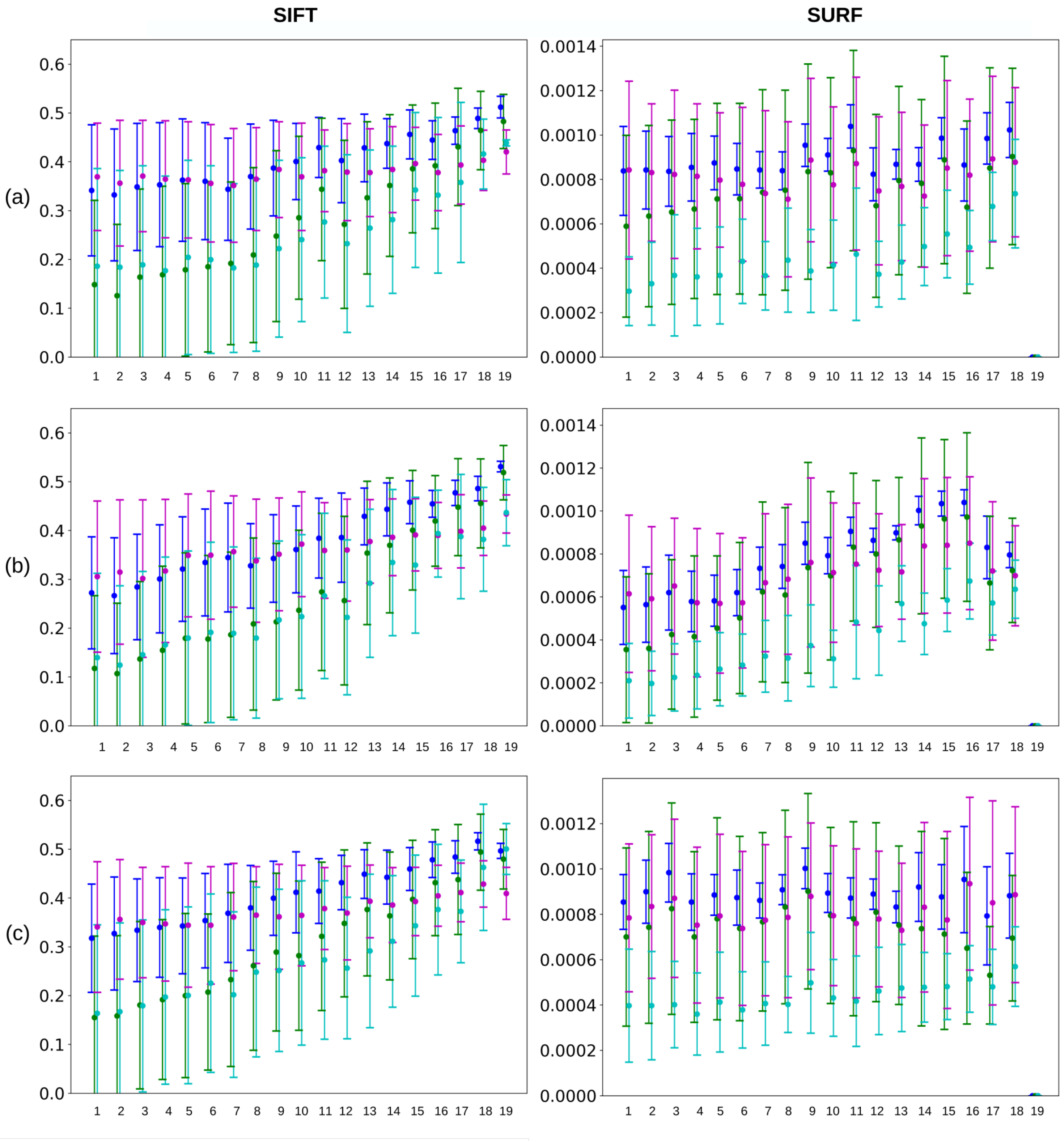


FIGURE 3.8: Tracking quality statistics for SIFT and SURF. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the normalized matching distance.

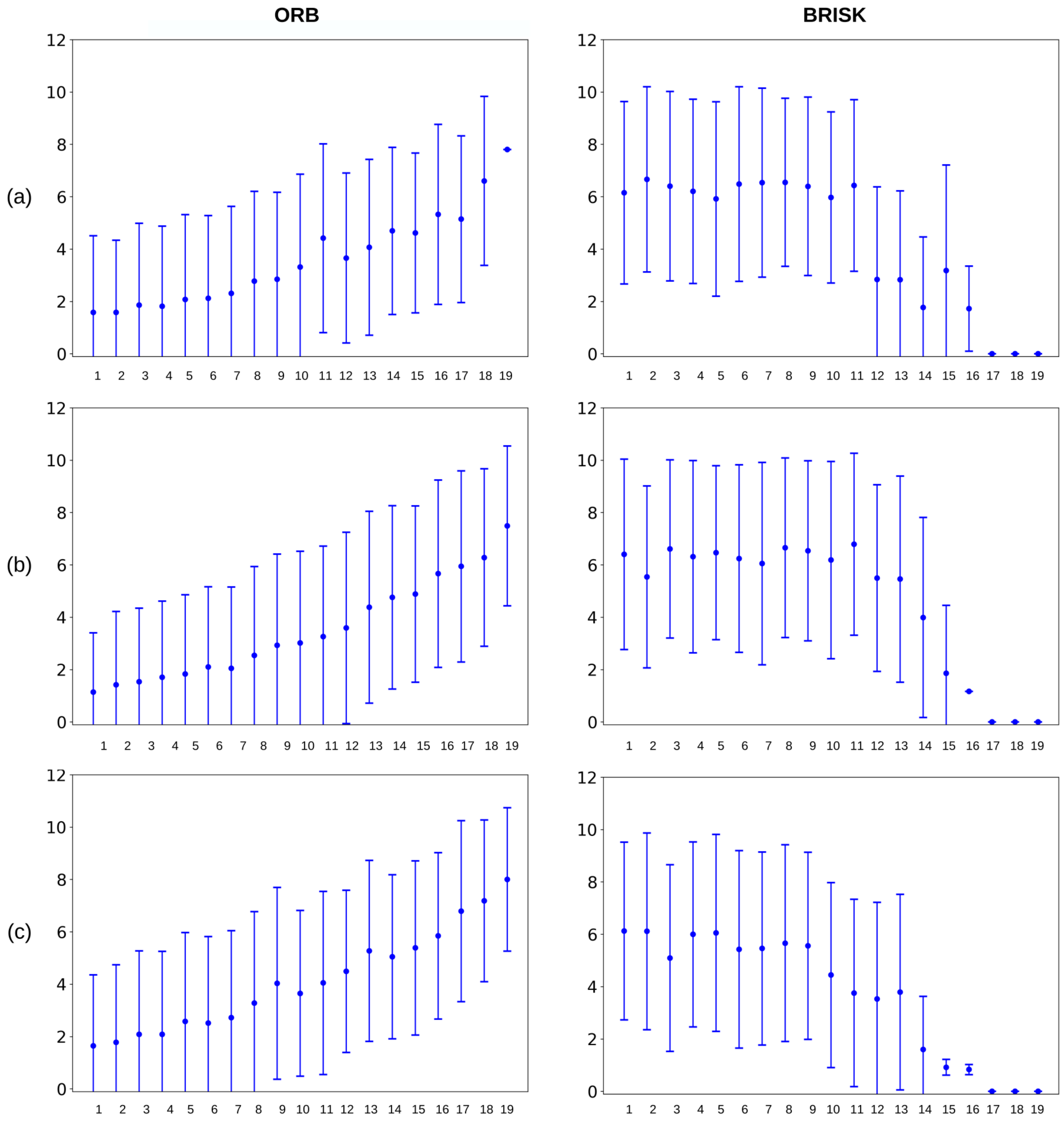


FIGURE 3.9: Tracking error statistics for ORB and BRISK. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the euclidean norm in pixel of the error.

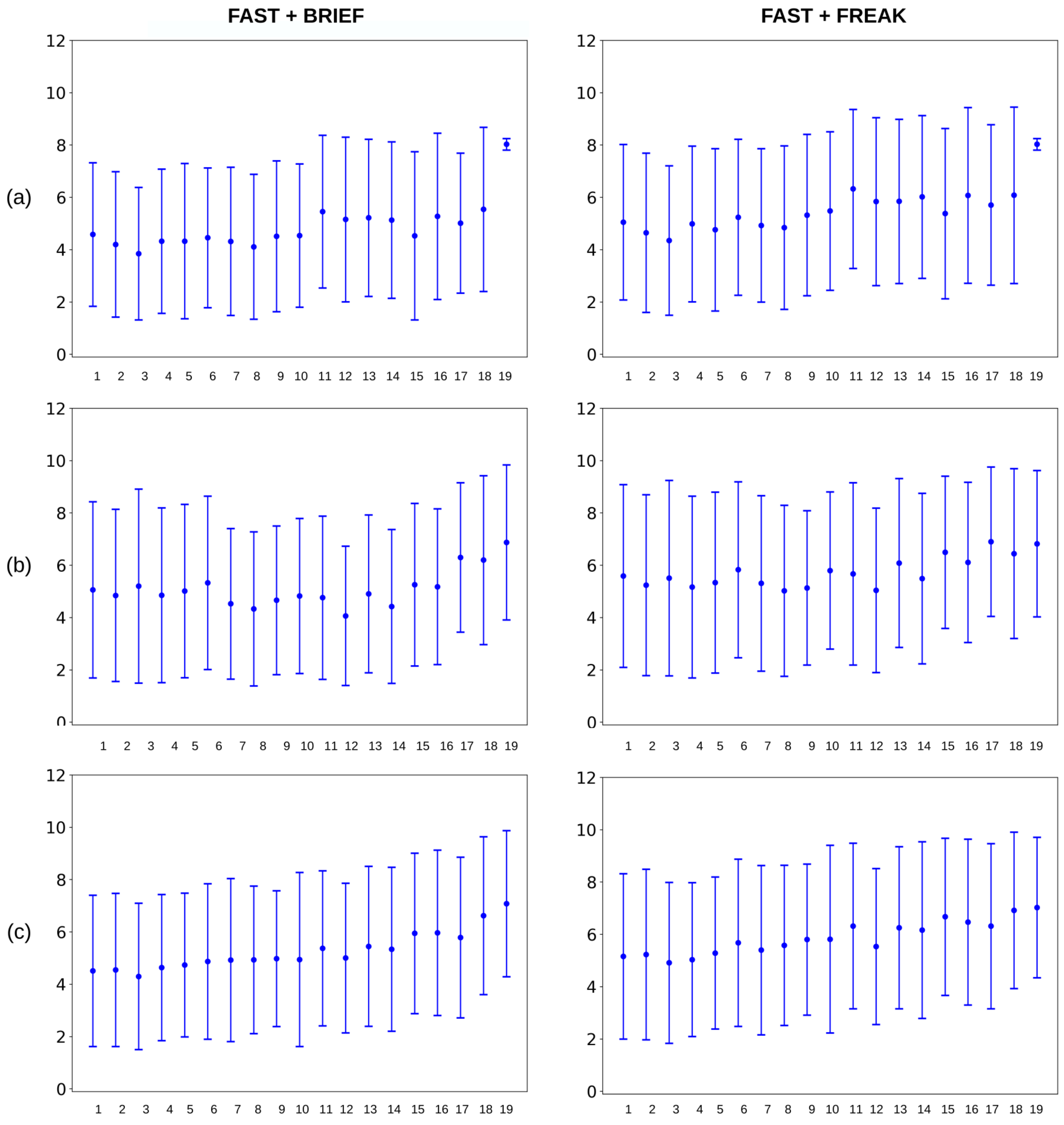


FIGURE 3.10: Tracking error statistics for FAST combined with BRIEF and FREAK. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the euclidean norm in pixel of the error.

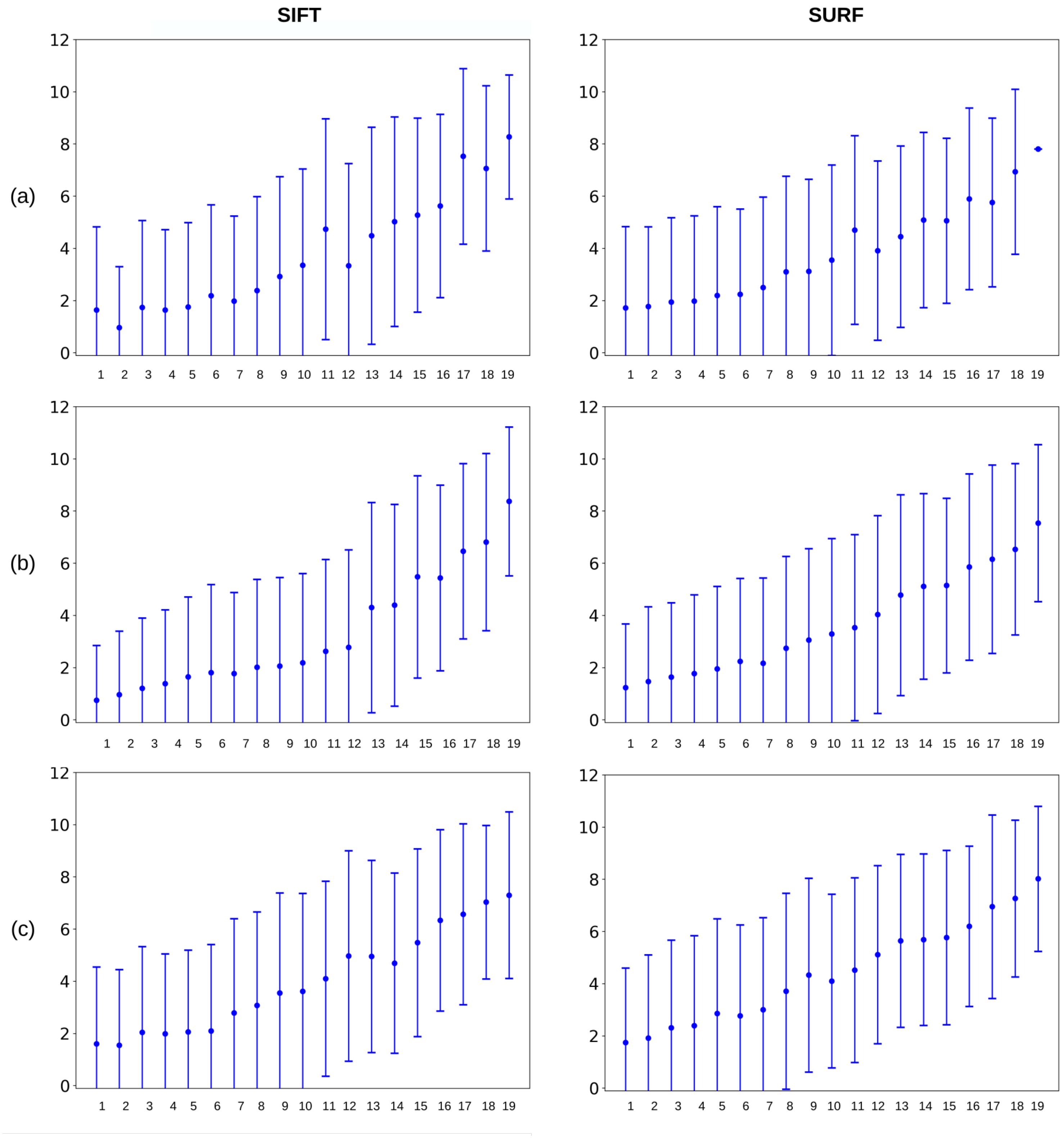


FIGURE 3.11: Tracking error statistics for SIFT and SURF. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the euclidean norm in pixel of the error.

this analysis the average distance values of correct matchings only, we expect to draw out a meaningful distance threshold that could be used to filter potential wrong matchings. Again, we expect this distance to be well distinguishable.

For both the indirect and direct methods, we also analyze their tracking error to evaluate their expected accuracy.

3.7.1.2 Tracking Quality Results Analysis

We now discuss the obtained results. First, we look at the tracking quality plots displayed in Fig. 3.6, 3.7, 3.8. As we can see, the general trend for all the methods is that the minimum distances are more and more difficult to identify as the turbidity increases. The same is true when we look at the good distances only. Hence, we expect these methods to perform quite well, as long as the turbidity remains below a certain level (approximately around the ninth image of the sets) but their efficiency quickly decreases past this level. The use of ORB, SIFT and SURF seems to give the best results as the good distance stays slightly lower than the others on most of the images.

If we now turn to the tracking error results displayed in Fig. 3.9, 3.10, 3.11, we can see that most of the methods perform poorly with an average error of 5 to 6 pixels when selecting the good matching for a feature as the one with the lowest distance. Only the ORB, SURF and SIFT methods perform better, with the average error of 2 to 4 pixels. This confirms the observation made on the tracking quality results. Note that, on the plots of Fig. 3.9-3.11, some spurious results sometimes appear on the last values. These are due to the fact that, with some detectors, a very low amount of features have been detected in the most turbid images. This also the reason why with BRISK the tracking errors seem to improve when turbidity increases.

We now compare these results to the ones obtained by computing the SSD. The tracking error statistics for this direct tracking method are displayed in Fig. 3.12. We can observe that the tracking errors are very low on the first three quarters of the sets (error $\leq$ 2 px) and get worse on the last quarter (around 4 to 5 pixels of error). If we now look at the KLT results on the same figure, we can see that the tracking accuracy is even better and barely exceeds 4 pixels of error in the highest level of turbidity. We can also note that, in the lowest levels of turbidity, the KLT results are very close of

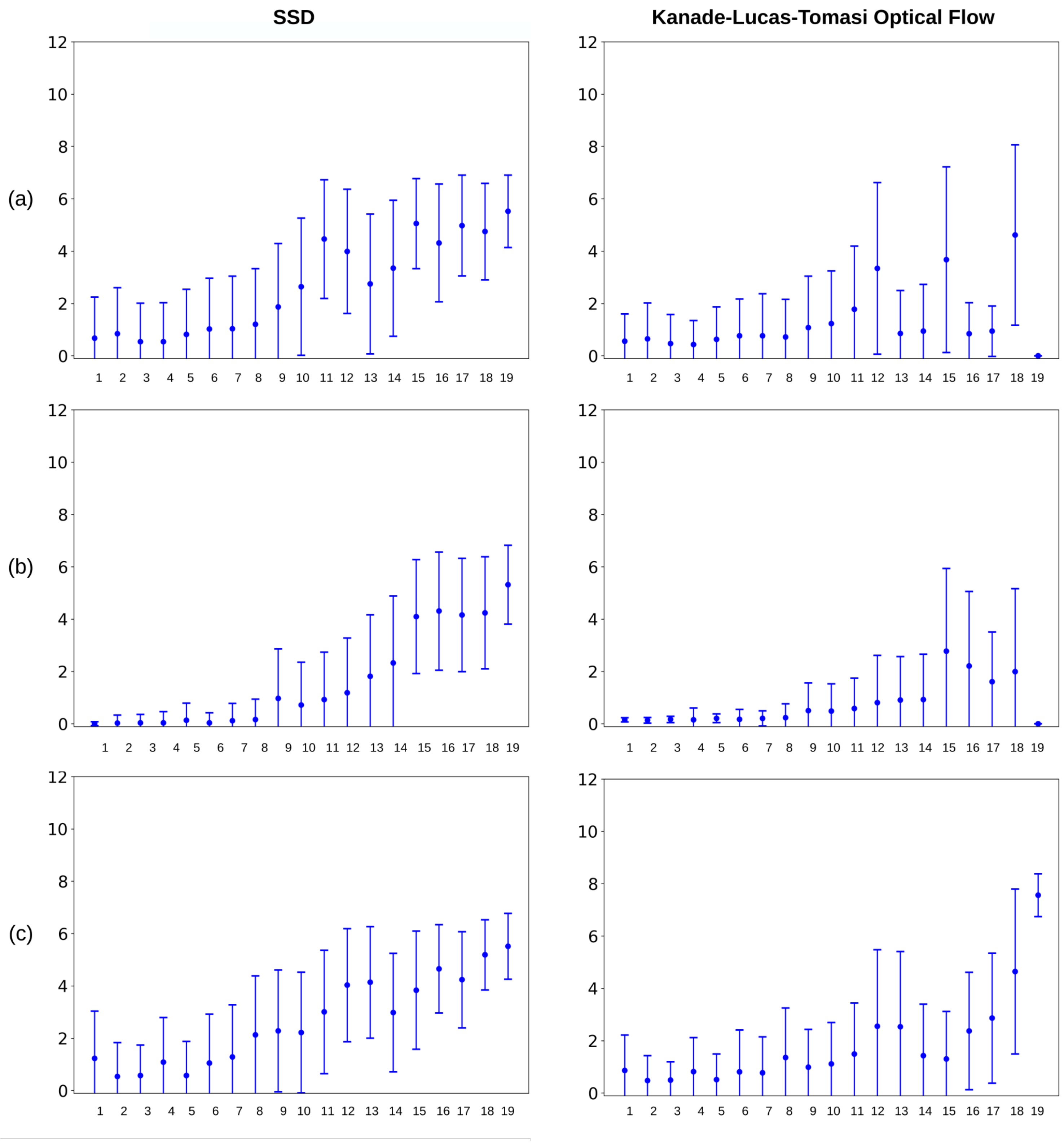


FIGURE 3.12: Tracking error statistics for the Sum of Squared Difference (SSD) method and the Kanade-Lucas-Tomasi (KLT) method. Each row displays the results obtained for the different sets of images: (a) TURBID #1, (b) TURBID #2, (c) TURBID #3. The y axis is the euclidean norm in pixel of the error.

those obtained with the SSD method. The reason for the improvements obtained with the KLT over the most turbid images lies in the fact that the KLT method rejects keypoints whose final solution are inaccurate (assessed by a final residual higher than a threshold).

From these first results, we can observe that the KLT method is the most accurate among the tested methods and ORB, SURF and SIFT seem to be more robust to the highest turbidity levels than the other descriptors.

3.7.2 Tracking Efficiency and Turbidity Robustness Analysis

We now look at the performance of the different methods in terms of the number of detected and correctly tracked features on the Turbid sets of images. The results are displayed in Fig. 3.13(a,b) which shows the evolution of the metrics for more and more turbid images. We recall that, here, the tracking is evaluated between pairs of successive images. It appears that the KLT method largely surpasses the indirect based ones in terms of the number of correctly tracked features. However, we can also see that, for most of the indirect methods, their detectors struggle in finding 500 well spread features over the images. On the other hand, the Shi-Tomasi detector used with the KLT is more efficient here. In order to remove any bias coming from a detector dependency, we run again the same analysis but using the Shi-Tomasi detector for every method. The results obtained are the ones shown in Fig. 3.13(c,d). The number of correctly tracked features increases impressively for every methods in this case. This is most likely due to a better repeatability of the Shi-Tomasi corner detector on this dataset. One can be surprised that the number of detections is different between the different methods whereas the same detector has been used. This is in fact due to the computation of the descriptors, which automatically removes features not suitable for a given descriptor. Nevertheless, we can notice that the KLT method still outperforms all the indirect methods on the first three quarters of the images sets and then performs comparably on the last quarter. This result is consistent with the tracking error analysis and highlights the robustness to mid-levels of turbidity of all the tested methods. Although, in front of very strong turbidity levels, no method seems to be really efficient.

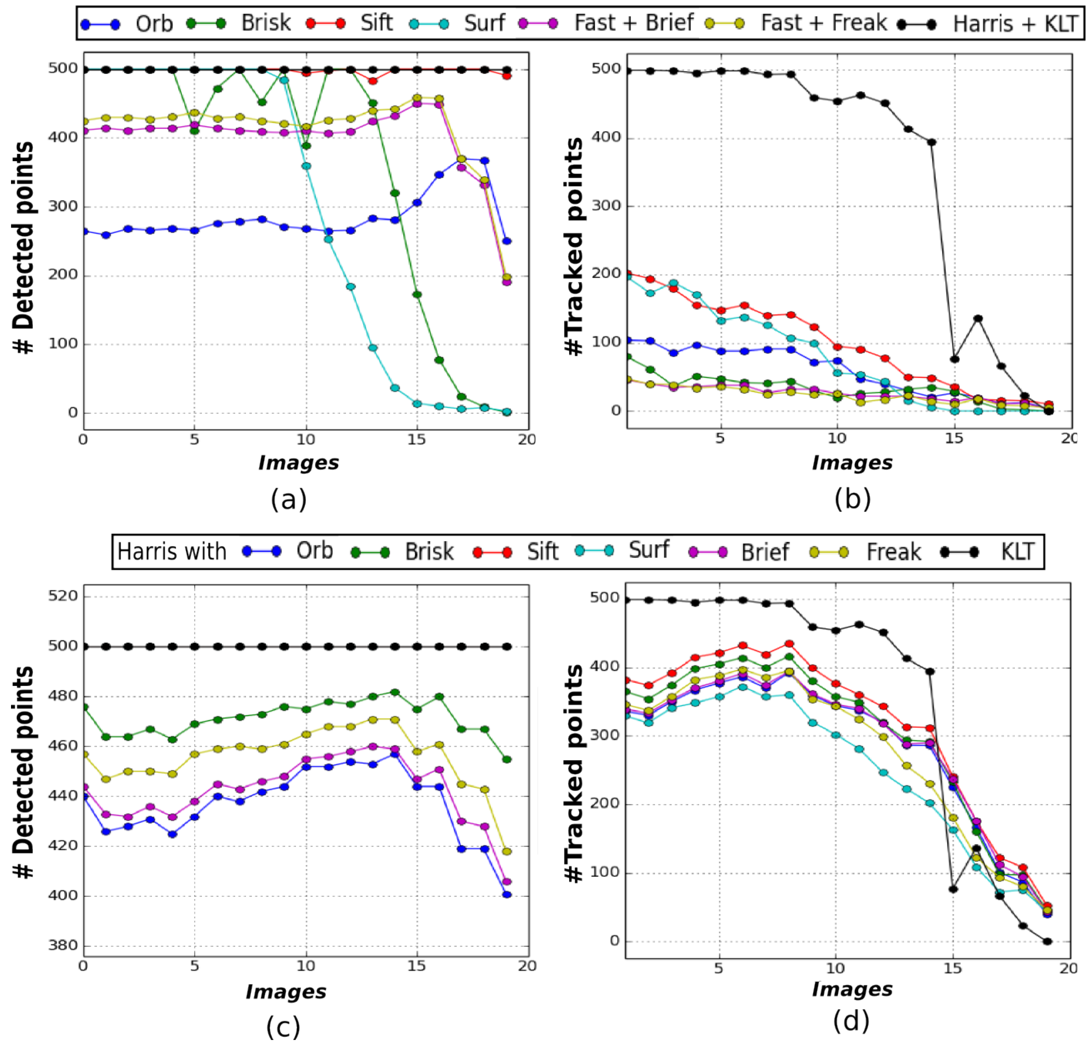


FIGURE 3.13: Tracking efficiency evaluation on the TURBID dataset [45] (presented in Fig. 3.4). Graphs (**a**) and (**b**) illustrates number of features respectively detected and tracked with different detectors while (**c**) and (**d**) illustrates number of features respectively detected with the Harris corner detector and tracked as before (the SURF and SIFT curves coinciding with the Harris-KLT one in (c)).

3.7.3 Tracking Efficiency and Low-Texture Robustness Analysis

We finally look at the performances obtained on the VO set. We recall that, on this set, we try to track features from the first image in all the following images – simulating a real VO scenario.

The results are displayed in Fig. 3.14(a,b). Once again the KLT method

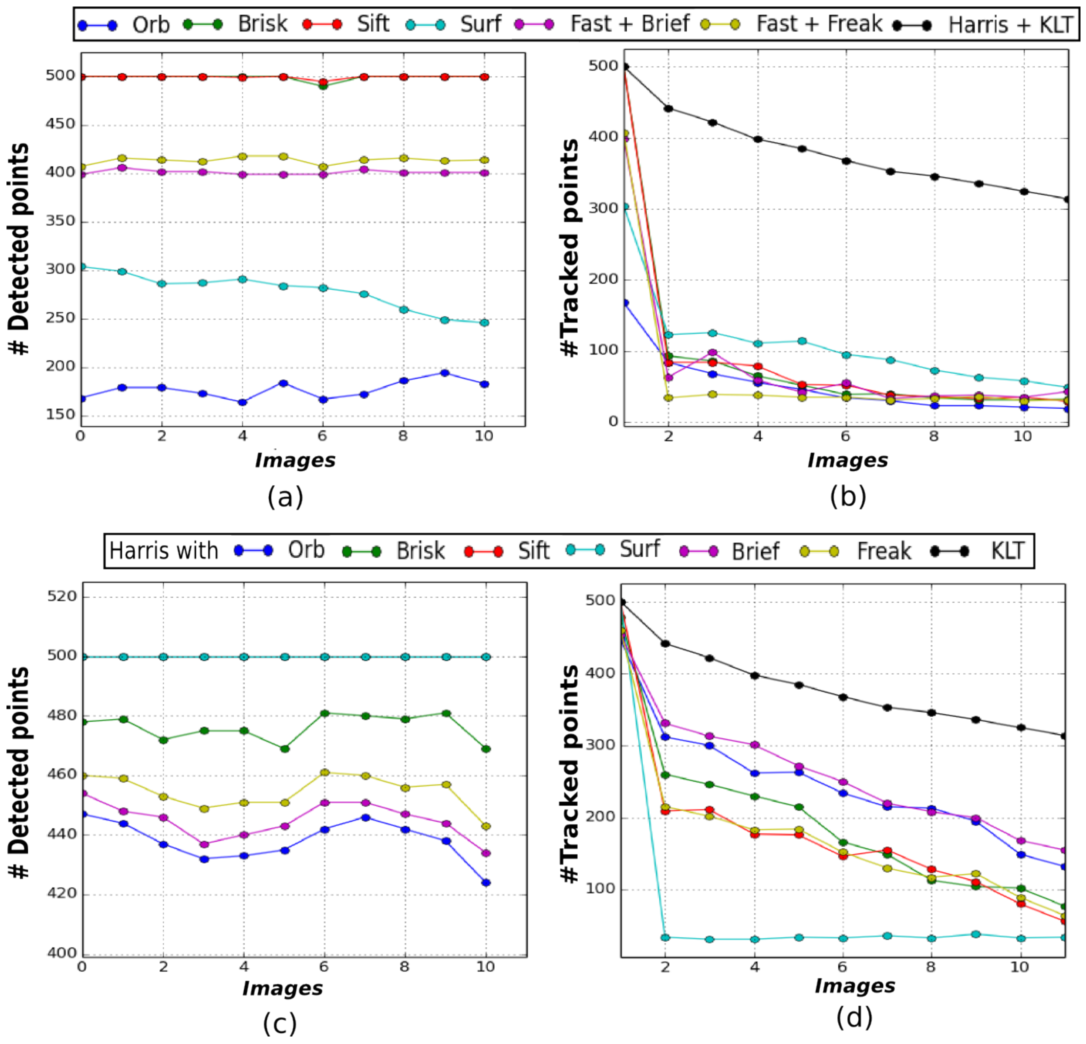


FIGURE 3.14: Tracking efficiency evaluation on the VO dataset (presented in Fig. 3.5). Graphs (**a**) and (**b**) illustrate the number of features respectively detected and tracked with different detectors, while (**c**) and (**d**) illustrate the number of features respectively detected with the Harris corner detector and tracked as before (the SIFT curve coinciding with the SURF one in (c)).

gives clearly better results. However, we also notice that the detectors employed with the indirect methods performs poorly on these low-textured images. Hence, we also run the same analysis using the Shi-Tomasi detector for all the methods. While this improves the performances of all the methods but SURF, it appears that the KLT method is more suited to the tracking task on these images. Besides, we can notice that the number of successfully tracked features with the ORB and BRIEF descriptors is quite high given the number of detected features in this case.

3.8 Conclusion

In this chapter, we have presented a thorough evaluation of the performance of different features tracking methods in an underwater context. We have compared indirect tracking methods, relying on the computation of descriptors as a mean of matching features, to direct tracking methods, based on the minimization of photometric errors. The conducted evaluation included the real-valued descriptors SIFT and SURF, the binary descriptors ORB, BRISK, BRIEF and FREAK (the last two used in combination with the FAST detector), the use of SSD and the Kanade-Lucas-Tomasi (KLT) sparse optical flow algorithm. We have focused this analysis on the efficiency of the different tracking methods in front of images with low-texture and varying levels of turbidity, which are typical characteristics of underwater imaging. The results obtained highlight the better performance of the KLT method in front of the indirect ones. This seems to be due to the fact that photometric minima are well defined until mid-levels of turbidity and in low-texture areas whereas the computed descriptors get more and more ambiguous.

In front of these results we conclude that the KLT method is well suited for the development of a visual localization system in underwater environment. It is still worth noting that the KLT method only detect features in the first image and then simply tracks them from one image to another. While this leads to a general lower computational cost of the KLT as new features do not have to be detected in each new image, it also means that any lost feature is definitely lost, even if only temporarily occluded. In opposition, the indirect methods, by extracting features in every images, have a way of finding temporarily invisible features. This can become an important flaw of the KLT if many occlusions occur in the acquired images. As underwater images sometimes suffer from such short occlusions due to the presence of moving animals, appealed by the embedded lighting systems.

As a perspective, it would be interesting to conduct a similar evaluation with recent deep learning based methods for features tracking, such as DELF [174] or Superpoint [51]. Also, a more robust version of the KLT method, partly due to the use of a robust estimator [105], has been recently added to OpenCV and would be worthy to test [198].

In the next chapter we will present a VSLAM method that builds upon the KLT method for the purpose of features tracking. Furthermore, the short occlusions issue will be taken into consideration.

Chapter 4

Robust Underwater Monocular Visual SLAM

Contents

4.1 Introduction

In this chapter we investigate the use of a monocular camera to perform simultaneous localization and mapping (SLAM). From the results obtained in the previous chapter, we propose a monocular Visual SLAM (VSLAM) method using the KLT (Kanade-Lucas-Tomasi) to perform features tracking through optical flow [21]. The method builds upon recent advances in VSLAM and is suited to underwater environments. The proposed VSLAM algorithm is *keyframe-based* and makes heavy use of Bundle Adjustment for optimizing the accuracy of the estimated trajectories.

Our contributions are the following:

- We propose UW-VO (UnderWater Visual Odometry), a real-time keyframe-based monocular VSLAM method that performs features tracking with the KLT method.
- We propose a features retracking mechanism to overcome the KLT weakness about lost features and gain in robustness.
- We show that the proposed method is more accurate and more robust than state-of-the-art monocular VSLAM methods on real-case underwater datasets.

This chapter is organized as follows. First, we review some of the major works in the monocular Visual SLAM field. Then, we expose formally the problem we are addressing. The proposed monocular VSLAM method is then detailed and experimental results are given to assess its efficiency.

4.2 Problem Statement

Visual SLAM (VSLAM) systems are localization solutions which estimate the trajectory followed by a camera from the images it acquires. The task of a VSLAM algorithm is to provide an estimation of the camera's pose at the camera's frame rate. Considering that a camera can move freely in a 3D space (6DOF), its pose can be modeled as an element of the 3D Special Euclidean group $\mathbb{SE}(3)$ composed of a rotational part $\mathbf{R}$, where $\mathbf{R}$ is a Rotation matrix and hence an element of the Special Orthogonal group $\mathbb{SO}(3)$, and a translational part $\mathbf{t} \in \mathbb{R}^3$.

The pose of the camera $\mathbf{X}_i$ at frame i can be expressed through its homogeneous representation:

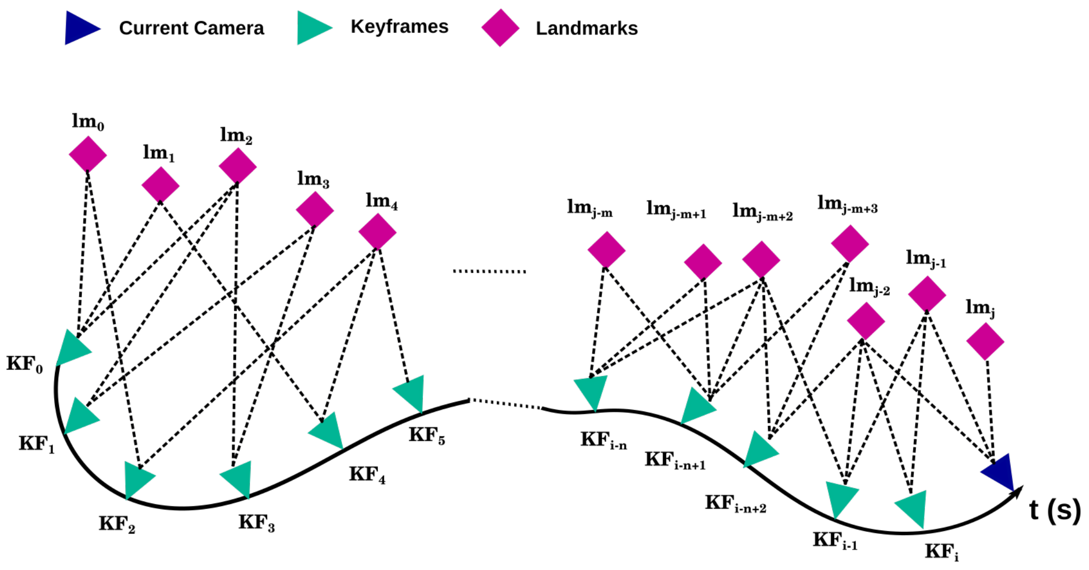


FIGURE 4.1: Illustration of the Visual SLAM problem. Keyframes are represented in turquoise and consist in informative past camera's states that are kept for the purpose of Bundle Adjustment. The map is composed of 3D landmarks (in pink) that are linked to keyframes through visual measurements (dotted lines). The current pose of the camera is linked to some of the most recent 3D landmarks.

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.1}$$

The matrix $\mathbf{X}_i$ here defines the transformation from the camera frame at frame i to a reference frame, often defined as the world frame w. To symbolize the transformation effects we use the following notations: $\mathbf{X}_{wc_i}$, to model the transformation from the camera frame to the world frame, and $\mathbf{X}_{c_iw}$ for the inverse transformation. These two transformations relate as:

$$\mathbf{X}_{wc_i} = \begin{bmatrix} \mathbf{R}_{wc_i} & \mathbf{t}_{wc_i} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.2}$$

$$\mathbf{X}_{c_iw} = \mathbf{X}_{wc_i}^{-1} = \begin{bmatrix} \mathbf{R}_{wc_i}^T & -\mathbf{R}_{wc_i}^T \cdot \mathbf{t}_{wc_i} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.3}$$

The localization problem in VSLAM is solved by comparing 2D visual measurements, $\tilde{\mathbf{x}} \in \mathbb{R}^2$, to 3D landmarks, ${}_w\mathbf{l} \in \mathbb{R}^3$. These 2D-3D correspondences relate through the projection functions $h(\cdot) : \mathbb{SE}(3) \times \mathbb{R}^3 \mapsto \mathbb{R}^3$ and $\pi(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^2$:

$$h\left(\mathbf{X}_{wc_i}, {}_w\mathbf{l}_j\right) = \mathbf{R}_{wc_i}^T \cdot \left({}_w\mathbf{l}_j - \mathbf{t}_{wc_i}\right) = \begin{bmatrix} x' & y' & z' \end{bmatrix}^T \tag{4.4}$$

$$\hat{\mathbf{x}}_j = \pi\left(\mathbf{K} \cdot h\left(\mathbf{X}_{wc_i}, {}_w\mathbf{l}_j\right)\right) = \begin{bmatrix} \frac{x'}{z'} \cdot f_x + c_x \\ \frac{y'}{z'} \cdot f_y + c_y \end{bmatrix} \tag{4.5}$$

$$\text{with } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

where $\hat{\mathbf{x}}_j$ is the pixel projection of the 3D landmark ${}_w\mathbf{l}_j$ mapped through the intrinsic calibration matrix $\mathbf{K}$ of the camera. From these projections, visual error terms can be defined as:

$$\mathbf{e}_{ij} = \tilde{\mathbf{x}}_{ij} - \hat{\mathbf{x}}_{ij} \tag{4.7}$$

where $\mathbf{e}_{ij}$ represent the reprojection error term obtained for the landmark ${}_w\mathbf{l}_j$ observed from the pose $\mathbf{X}_{wc_i}$.

In this thesis, the developed VSLAM system is *keyframe-based*, meaning that, in addition to the current pose of the camera, we also continuously estimate the poses of keyframes along with the 3D landmarks observed by these keyframes. The full states of interest at frame i are therefore:

$$\chi_i = \begin{bmatrix} \mathbf{X}_{wc_i} & \mathbf{X}_{wk} & {}_w\mathbf{l}_m \end{bmatrix}^T \tag{4.8}$$

where $\mathbf{X}_{wk}$ is the set of previously selected keyframes and ${}_w\mathbf{l}_m$ the set of estimated 3D landmarks. An illustration of the full VSLAM problem under the form of a graph is given in Fig. 4.1.

4.3 Related Works

Monocular SLAM has been first addressed with filtering methods [57, 49, 43]. In these works, an Extended Kalman Filter is used to jointly estimate the pose of the the camera along with the 3D landmarks at every frame. However, the EKF is not well suited to handle such large state vector and, because of the high nonlinearity of the VSLAM problem, many linearization errors are integrated. The application of these filtered methods have therefore been limited to small areas.

In opposition to filtered methods, keyframe-based methods have been proposed. The idea behind keyframes is that using every acquired frames for estimating 3D landmarks has little interest, because of high redundancy in the measurements, and only selecting non-redundant frames (*i.e.* keyframes) for the mapping steps ends up being more efficient and more accurate. Furthermore, the subsampling implied by the keyframes paradigm provides a way of efficiently using Bundle Adjustment optimization [216] in real-time. Bundle Adjustment is a nonlinear least-squares optimization which aims at estimating both cameras' poses and 3D landmarks by minimizing visual reprojection errors. Bundle Adjustment is more costly than a single iteration in an EKF but provides better estimates as it iteratively refines its estimations [211]. PTAM [125] was the first monocular SLAM algorithm able to handle Bundle Adjustment in real-time. Their main idea, besides making use of keyframes, was to separate the tracking and mapping part of the VSLAM algorithm in two different threads. Hence, their method efficiently uses Bundle Adjustment in the mapping thread while the tracking thread takes care of features tracking and pose estimation at frame rate. Even if the Bundle Adjustment operation takes longer than the processing of one frame in the tracking thread, the algorithm is not impacted and still benefit from it (thanks to the optimization of 3D landmarks currently observed).

Since PTAM, many improvements have been proposed to address monocular VSLAM. In [210], a dense tracking based on optical flow and implemented on GPU was proposed. In addition, SURF features [15] were extracted for loop-closure detection and they proposed to use 7DOF similarity constraints (6DOF for the pose and 1DOF for the scale) to optimize the resulting pose-graph and recover from potential scale drift in the monocular estimations. The same author proposed, in [209], an improved version of PTAM by adding loop-closure detection and pose-graph optimization within the mapping thread. In [207], a Visual Odometry method inspired by PTAM and targeting autonomous car navigation was proposed. Their method uses ORB features [191] for fast tracking and several modifications of the algorithm architecture were applied to optimally run on multicore CPUs. Then, in [144], a monocular VSLAM able to perform tracking, mapping and loop-closure in real-time was proposed. The algorithm is largely inspired by PTAM and uses BRIEF features [26] for both tracking and loop-closure. At the same time, ORB-SLAM [161] was proposed. This algorithm is also able to perform tracking, mapping and loop-closure in real-time. However, it uses ORB features instead of BRIEF, with better relocalization properties. ORB-SLAM has been considered as the state-of-the-art feature-based monocular VSLAM so far and its open-source release has made it very popular.

In parallel to feature-based methods, that handle the VSLAM problem by the tracking of features, direct methods have been emerging recently. Direct methods solve the SLAM problem by performing localization and tracking in a joint optimization, directly using the pixels intensities to define the associated cost function. Direct methods can be dense, DTAM [167], meaning that all the image's pixels are used, semi-dense, LSD-SLAM [62], if only pixels with strong gradients are used, or sparse, DSO [63], if only a small set of pixels is used. The advantage of direct methods is that they do not rely on detected features but directly leverage the pixels' intensity for the localization and mapping task. They are expected to be more robust to scenes with low texture as they do not require strong distinctive features as feature-based methods do. However, they are computationally more demanding and not as mature as feature-based methods yet. A so-called semi-direct method [79, 81] has also been proposed. It combines the accuracy of sparse direct methods with the robustness of features detector to initialize the set of sparse pixels to use for localization and mapping. This method was shown to be extremely fast but also works better with high frame-rate cameras.

With respect to these works, our monocular VSLAM method is similar to recent feature-based methods. However, the features tracking part does not depend on descriptors but on a direct tracking method based on optical flow. This relax the constraint of extracting features in every image. Instead, features are only detected in keyframes and tracked frame-to-frame using their photometric appearance. To the best of our knowledge, there is no previous keyframe-based monocular VSLAM method that has been proposed for underwater localization.

4.4 Overview of the Visual SLAM algorithm

The pipeline of UW-VO is summarized in Fig. 4.2. The system is based on the tracking of 2D features over successive frames in order to estimate their 3D positions in the world referential. The 2D observations of these 3D landmarks are then used to estimate the motion of the camera. Frames used for the triangulation of the 3D map points are considered as keyframes and the most recent ones are stored in order to optimize the estimated trajectory along with the structure of the 3D map through bundle adjustment. The method follows the approach of [124, 208, 161]. However, in opposition to these methods, we do not build the tracking on the matching of descriptors.

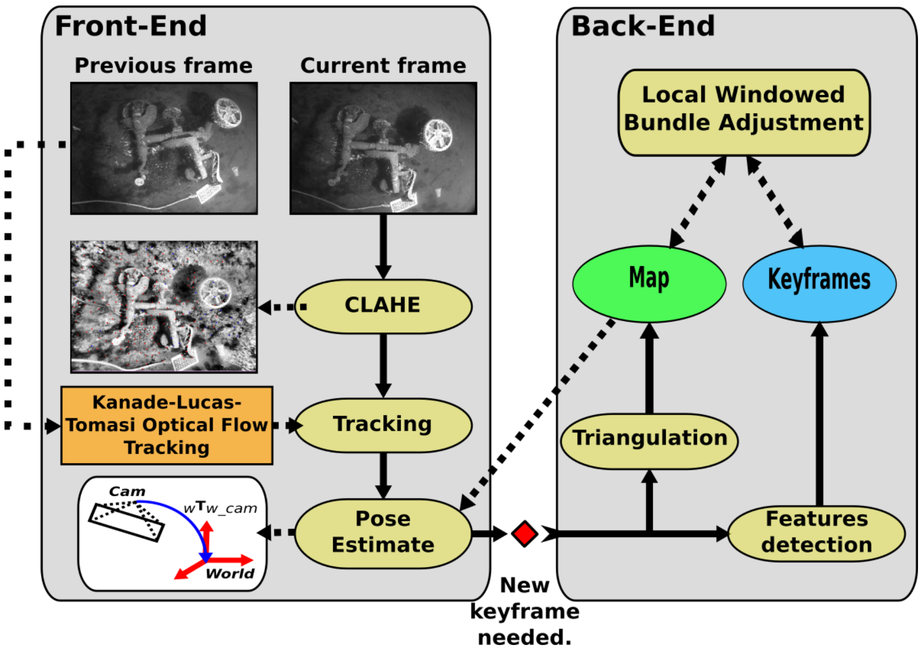


FIGURE 4.2: Pipeline of the proposed Visual SLAM algorithm UW-VO. The front-end thread is responsible for the tracking and pose estimation at frame-rate while the back-end thread takes care of the mapping and Bundle Adjustment related operations.

Instead we use the KLT method, more adapted to the underwater environment as demonstrated in the previous chapter. The drawback of the KLT in opposition to descriptors is that it is only meant for tracking features between successive images. This is a problem when dealing with a dynamic environment as good features might be lost because of short occlusions. To make the KLT robust to such events, a retracking mechanism is added to the tracking part of the VSLAM framework. This mechanism will be described in section 4.5.4.

As illustrated in Fig. 4.2, the algorithm is divided into two threads. A front-end thread is responsible for the tracking part and the pose estimation at frame-rate. In parallel, the back-end thread takes care of the operations related to the creation of new keyframes, that is mapping and Bundle Adjustment. The front-end thread outputs a real-time estimation of the camera's pose while the back-end thread performs the more complex optimization tasks.

We first describe the processing performed within the front-end thread and then we describe in-depth how the back-end thread handles the Bundle Adjustment operations.

4.5 Visual SLAM Front-End

4.5.1 Handling Image distortion

The visual measurements are formed by the detection and tracking locations of specific keypoints. These measurements can hence directly be modeled by their pixel coordinates in the image: $\mathbf{x} = \begin{bmatrix} u & v \end{bmatrix}^T$, $\mathbf{x} \in \mathbb{R}^2$. However, the processed images always suffer from lens distortion effects which invalidate most of the multi-view geometry properties, defined for pinhole cameras with perfect lenses. The undistorted coordinates of the keypoints can be obtained by applying an undistortion function $\gamma(\cdot)$: $\mathbf{x}_{undist} = \gamma(\mathbf{x})$. The function $\gamma(\cdot)$ is a polynomial function reflecting the lens geometry. These distortion effects can be estimated through a calibration step along with the intrinsic parameters of the camera's sensor [228, 229].

To avoid the need of applying this undistortion function to every visual measurement, a classical solution is to compute a 2D mapping of the pixel positions between the distorted and undistorted image. The distortion effects being constants, this mapping can hence be computed once and then applied to all the images acquired by the camera. The advantage is that the visual measurements directly correspond to their undistorted coordinates. However, the distortion removal process often leads to some cropping in the original image.

The downside of cropping all the images is the loss of visual information, thus reducing the keypoints tracking boundaries. In order to keep using the full image, the visual measurements can also be modeled as bearing vectors as represented in Fig. 4.3. This way the keypoints are defined as undistorted 3D unit-length vectors originating from the optical center of the camera. This representation can be used to directly solve the classical multi-view geometry algorithms, without any constraints over the cameras' models used. In practice, the bearing vectors are computed as follow:

$$\mathbf{bv}(\mathbf{x}) = \frac{\mathbf{x}^c}{\|\mathbf{x}^c\|^2} \text{ , with } \mathbf{x}^c = \mathbf{K}^{-1} \cdot \begin{bmatrix} \gamma(\mathbf{x}) \\ 1 \end{bmatrix} \tag{4.9}$$

This representation is the one used in UW-VO and hence makes it compliant with any camera models, as long as the undistortion function is given. Furthermore, the use of bearing vectors allows to model easily multi-cameras systems [128]. The developed algorithm could therefore be extended for such configurations.

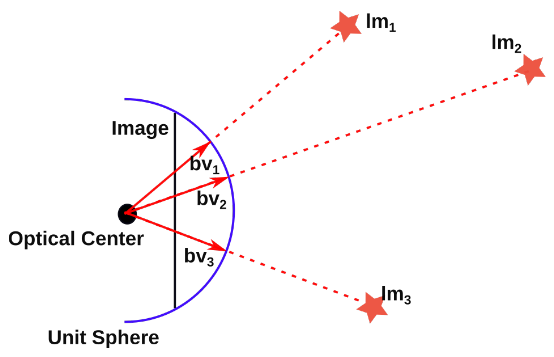


FIGURE 4.3: Bearing vectors representation. The undistorted keypoints are represented as unit vectors $\mathbf{bv}_i$, projected onto the unit sphere around the optical center of the camera and and pointing towards their landmarks $\mathbf{lm}_i$.

4.5.2 Image Pre-Processing

The poor imaging conditions of underwater environments often lead to the acquisition of images with a lack of contrast. This might be due to the lack of texture of the imaged scenes but also to shadows or overexposed areas created by the embedded lighting system.

As a minimum of contrast in the images is required for the tracking and detection of keypoints, we pre-process the acquired images before using them in UW-VO. This pre-processing applies a *Contrast Limited Adaptive Histogram Equalization* (CLAHE) [232] technique in order to enhance the contrast in the images. This method is somewhat similar to the classical Histogram Equalization (HE) method but, instead of performing it onto the full image, it applies HE locally, on patches moved over the image. By increasing the contrast of patches, based on the patches' intensity dynamic, we ensure a better local contrast enhancement than with classical HE, which actually might decrease the contrast of some image regions in order to increase the global contrast of the image. The price to pay for this contrast enhancement is an increase of noise in the image. However, the CLAHE algorithm reduces this tendency by clipping the intensities histogram in order to keep an homogeneous spreading of intensities over the patches. This is especially useful with homogeneous patches, which become very noisy if no clipping is used. The effect of this contrast enhancement can be seen within the pipeline figure of UW-VO in Fig. 4.2.

Despite the slight increase of noise, this pre-processing tends to improve the VSLAM algorithm by making the tracking and the detection of keypoints more robust to a lack of contrast and to overexposure.

4.5.3 Frame-to-Frame Features Tracking

The tracking of features is done frame to frame, that is from the image $\mathbf{I}_{t-1}$ to the image $\mathbf{I}_t$. The tracking is performed through the estimation of the optical flow using the KLT method [21] described in chapter 3.

In order to increase the robustness of the tracking and remove outliers as soon as possible, this tracking is performed in a forward-backward scheme. The estimation of the optical flow is therefore computed two times: first from $\mathbf{I}_{t-1}$ to $\mathbf{I}_t$ and then from $\mathbf{I}_t$ to $\mathbf{I}_{t-1}$. The error between the original keypoints position and the one found in the backward step is used to validate or not the tracking of each keypoint. In practice, we consider any keypoint with an error of more than 1 pixel as being ambiguous and remove it from the set of tracked features.

The removal of outliers is extended further by checking the epipolar geometry consistency of the currently tracked features. This is done by computing the Essential Matrix between the last keyframe and the current frame (see sec. 2.3.2). The Essential Matrix $\mathbf{E}$ is defined as follow:

$$\mathbf{x}' \cdot \mathbf{E} \cdot \mathbf{x} = 0 \tag{4.10}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the homogeneous coordinates of the observations of a same 3D landmark in two different images. Using the method of Nister [172], $\mathbf{E}$ can be computed from 5 pairs of keypoints. However, in general, we will track many more keypoints and a unique solution will not exist because of noise in the observations and because of the potential presence of outliers. In order to detect and remove such outliers, $\mathbf{E}$ is computed in a RAndom SAmple Consensus (RANSAC) scheme. RANSAC is a robust parameters estimation mechanism able to detect outliers from a set of sample, described in section 2.3.4 (page 27).

This tracking step ensures that only reliable features will enter in the next stages of the algorithm.

4.5.4 Features Retracking

As pointed out in chapter 3, the main drawback of optical flow tracking is that lost features are usually permanently lost. In opposition, the use of descriptors gives a way of retracking lost features. In the underwater context, the powerful lights embedded by ROVs often attract schools of fishes or shrimps in the camera's field of view (see Fig. 4.4). The occlusions due to animals can lead to strong photometric shifts and consequently to a quick

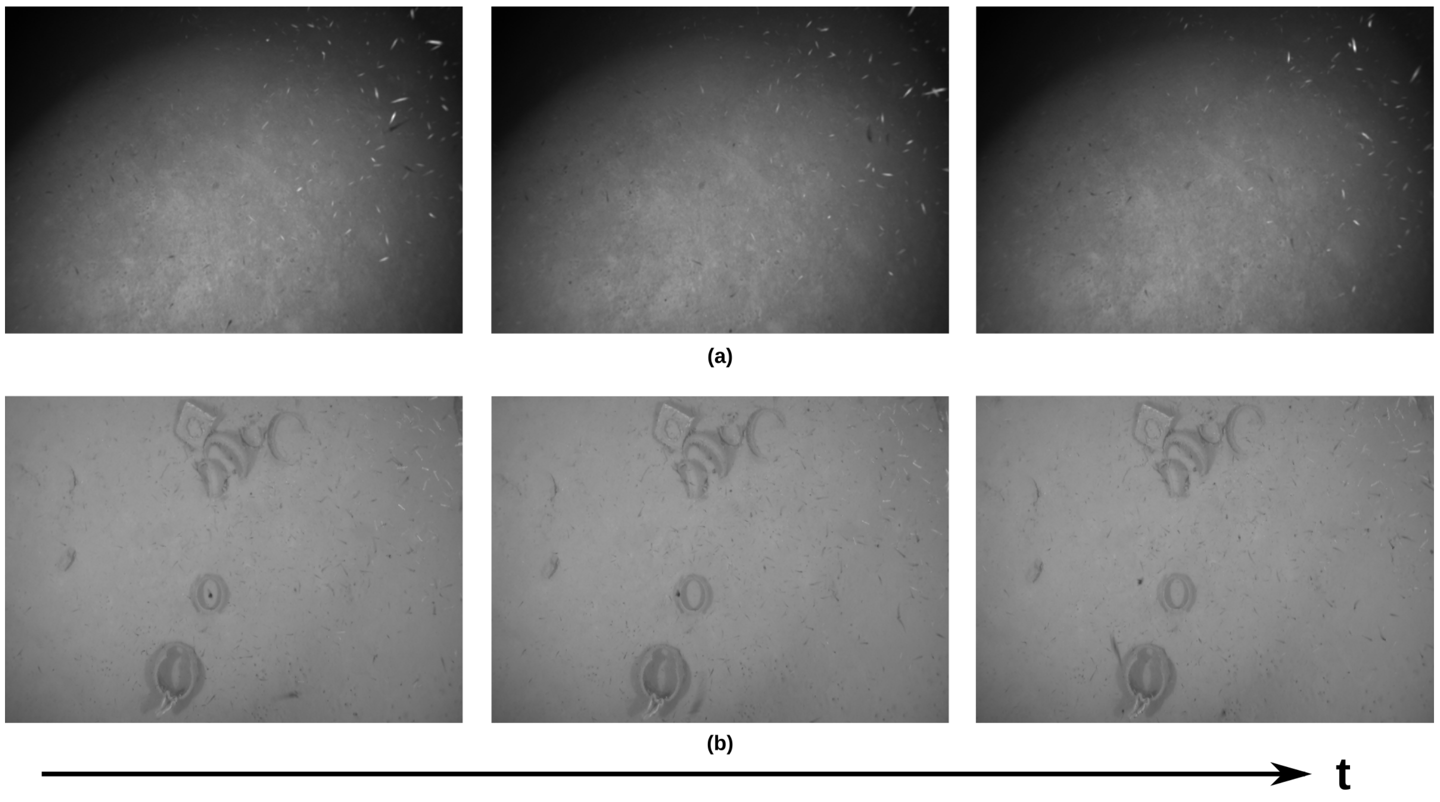


FIGURE 4.4: Sequence of images disturbed by moving fishes. Better seen in pdf format when zooming.

loss of features. However, fishes are moving very fast in comparison with the camera motions. We take advantage of this fact to increase the robustness of our tracking method over short occlusions. The employed strategy is too keep a small window of the most recent frames (five frames are enough in practice) with the features lost through optical flow in it. At each tracking iteration, we try to retrack with the KLT the lost features contained in the retracking window. Finally, retracked features are added to the set of currently tracked features. This features retracking mechanism is used to retrack both pure 2D features, for future triangulation, and 2D observations of already mapped points, for pose estimations.

4.5.5 Pose Estimation

The estimation of the 6DOF of the pose of every frame uses their respective 2D-3D correspondences. The pose is computed with the Perspective-from-3-Points (P3P) formula (presented in sec. 2.3.3), using the algorithm of [127]. P3P requires a total of four 2D-3D observations to estimate the pose of the camera, three to compute a first set of hypotheses and the last one to disambiguate and find the right solution within these hypotheses.

This operation is also done within a RANSAC loop to remove inaccurate correspondences. The pose of the current camera $\mathbf{X}_i$ is hence computed from the combination of points giving the most likely estimation for the set of features.

In order to improve the accuracy of the estimated pose, we further refine it through the minimization of the global reprojection error in the image. This is done by minimizing the following Mahalanobis distance:

$$\mathbf{X}_i^* = \underset{\mathbf{X}_i}{\arg\min} \sum_{j \in \mathfrak{L}_i} \left\| \mathbf{x}_{ij} - \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) \right\|^2_{\mathbf{\Sigma}_{visual}} \tag{4.11}$$

where $\mathfrak{L}_i$ is the set of landmarks observed from $\mathbf{X}_i$, $\mathbf{x}_{ij}$ is the keypoint corresponding to the observation of the landmark ${}_w\mathbf{l}_j$ in $\mathbf{X}_i$ and $\mathbf{\Sigma}_{visual}$ is the covariance associated to the visual measurements.

This minimization is done through a nonlinear least-squares optimization using the Levenberg-Marquardt algorithm. The detail of how this optimization is performed are not given here but will be described in depth in section 4.6.

4.5.6 Keyframe Selection and Mapping

The developed VSLAM algorithm is keyframe-based, meaning that only a subset of the processed frames are used for the mapping stage of the algorithm. Moreover, these keyframes are also stored for the purpose of continuously optimizing the estimated trajectory and 3D map. By merely using keyframes instead of every frame for optimization, the computational load is highly reduced and very redundant measurements are discarded to the benefit of more distinctive ones.

If the creation of a new keyframe is triggered, the current frame is stored and new 3D landmarks are estimated from the pure 2D keypoints tracked from the previous keyframe. New features to track are then detected within the current image using the Shi-Tomasi detector [215, 199]. Last, this new keyframe is sent to the back-end thread of the algorithm along with the newly mapped 3D landmarks for a multi-view optimization through a Bundle Adjustment operation.

The mapping process is triggered by the creation of new keyframes. Several criteria have been set as requirements for the creation of a keyframe. The first criterion is the parallax. If an important parallax from the last keyframe has been measured (30 pixels in practice), a new keyframe is inserted as it will allow the computation of accurate 3D landmarks. The parallax is estimated by computing the median disparity of every tracked pure 2D features from the previous keyframe. To ensure that we do not try to estimate 3D landmarks from false parallax due to rotational motions, we unrotate the currently tracked features before computing the median disparity. The second criterion is based on the number of 2D-3D correspondences. We ensure that we are tracking enough 3D landmarks and trigger the creation of a keyframe if this number drops below a threshold.

4.5.7 Initialization

Monocular systems are subject to a "Chicken and Egg" problem at the beginning. Indeed, the motion of the camera is estimated through the observations of known 3D world points, but the depth of the imaged world points (*i.e.* the real distance between the 3D point and the camera's optical center) is not observable from a single image. The depth of these world points can be estimated using two images with a sufficient baseline (*i.e.* taken from different positions). However, this baseline needs to be known to compute the depth and vice-versa. This is why monocular VSLAM requires an initialization step to bootstrap the algorithm in opposition to stereoscopic systems where the baseline is fixed and known (from a calibration step).

Here, we propose to initialize the algorithm by computing the relative pose between the current frame and the first keyframe as soon as enough parallax is detected. This relative pose is estimated from the Essential Matrix **E**:

$$\mathbf{E} = \mathbf{R} \cdot \mathbf{t}^{\wedge} \tag{4.12}$$

where the $(\cdot)^{\wedge}$ operator turns a 3D vector into a skew-symmetric matrix. Four possible solutions for **R** and **t** can be extracted from **E**, but only one is realistic and results in a motion that produces 3D landmarks in front of both cameras (the three others solutions leading to solutions that would produce 3D landmarks behind one or both of the cameras).

The relative pose between both keyframes, up to a scale factor is recovered, from this valid solution. The actual length of the translation vector cannot be recovered because of the scale invariance and is thus arbitrarily fixed. The resulting relative pose is used to bootstrap the algorithm by estimating a first set of 3D landmarks. Note that this initialization step fixes the scale of the trajectory, as the depth of the first set of 3D landmarks is derived from the chosen norm of the translation.

4.6 Visual SLAM Back-End

The back-end thread of the VSLAM algorithm is responsible for the optimization of the current trajectory. By receiving every keyframes along with their observed 3D landmarks, it builds a minimization problem that applies multi-view constraints to the estimated 3D landmarks from their observing keyframes. The minimization is performed over the visual reprojection errors and allows to find an optimal configuration between the keyframes poses and the 3D landmarks. This process is referred to as Bundle Adjustment in the literature [216].

4.6.1 The Bundle Adjustment Problem

Bundle Adjustment is a technique, first developed in the Photogrammetry field, that aims at simultaneously optimizing the poses of cameras and the 3D landmarks observed by these cameras. This optimization is performed by minimization of the reprojection errors and can be seen as a way of adding multi-view constraints on the 3D landmarks. If no prior is provided (besides an initial solution to the nonlinear problem), the Bundle Adjustment operation can be defined as a Maximum Likelihood Estimation (see sec. 2.5), where the states of interest are the cameras' poses and the 3D landmarks, which are linked by visual measurements (the respective observations of the landmarks into the cameras' images).

The Bundle Adjustment can therefore be used as a full smoothing estimator, where all the states of interest are optimized, or as a fixed-lag smoothing estimator, where only a subset of the most recent states are optimized [196].

The Bundle Adjustment problem is written as a Non-Linear Least Squares (NLLS) problem where one wants to minimize the Mahalanobis distance over the set of reprojection errors:

$$\chi^* = \arg\min_{\chi} \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \left\| \mathbf{x}_{ij} - \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) \right\|^2_{\mathbf{\Sigma}_{visual}} \tag{4.13}$$

where is χ is the state vector, $\mathfrak{K}$ denotes the set of keyframes, $\mathfrak{L}_i$ the set of landmarks observed from these keyframes and $\mathbf{\Sigma}_{visual}$ is the covariance over the visual measurements.

A direct way for solving this problem would be to apply a Gauss-Newton like method to estimate the optimal keyframes states $\mathbf{X}_{wk_n}$ and landmarks positions ${}_{w}\mathbf{l}_j$. However, NLLS solvers expect that the states which are to be optimized belong to a pure Euclidean space. This is not the case here because the keyframes states, being elements of $\mathbb{SE}(3)$, evolves on a manifold (see sec. 2.7).

The problem when using classical NNLS optimization with states defined on a manifold is that the optimization will be performed without accounting for the special structure of this manifold. In other words, the optimizer will come up with a solution that might violate the properties of a manifold. For example, optimizing rotation matrices, which are elements of $\mathbb{SO}(3)$, would most likely update the coefficients that define these matrices in a way such that the optimized matrices will not be element of $\mathbb{SO}(3)$ anymore (for example the matrices orthogonality might disappear). Moreover, elements of manifolds are often represented with more parameters than degrees of freedom. For instance, a rotation matrix has 9 parameters for only 3DOF. Therefore, optimizing these 9 coefficients would be suboptimal and would lead to an excessive complexity.

The correct way of handling NNLS optimization over elements defined on manifolds is to use their associated Lie algebra [103, 206]. Lie algebra defines a *local* parametrization for manifolds that is Euclidean. This local parametrization is generally called the tangent space. Through this Euclidean tangent space, Lie algebra provides a tool for optimizing states defined on a manifold. The general idea is that the optimization does not have to be performed over the full parametrization of the states anymore, but only over their local parametrization. Such optimizations are built upon a *lift-solve-retract* scheme [1], which allows the optimization to be performed on the Euclidean tangent spaces while keeping the manifold structure of the states of interest.

In our case, the optimization has to be done over the $\mathbb{SE}(3)$ manifold (which defines the cameras' poses). Such parametrization leads to the following modifications in the NLLS solving approach:

- The optimization has to be performed within a *lift-solve-retract* scheme in order to compute the small perturbations to apply to the states in the Euclidean space defined by the Lie Algebra of the manifold ($\mathfrak{se}(3)$ in our case).

- Defining *box-plus* $\boxplus$ and *box-minus* $\boxminus$ operators which adapts the classical vector addition and subtraction to the manifold elements.

4.6.2 On-Manifold Optimization

$\mathbb{SE}(3)$ is a 6-dimensional manifold with the manifold structure $\mathbb{SO}(3) \times \mathbb{R}^3$, where $\mathbb{SO}(3)$ stands for the 3D Special Orthogonal group manifold defining 3D rotation matrices. Any elements of $\mathbb{SE}(3)$ is thus composed of 12 parameters, 9 defining the rotation matrix and 3 defining the translational component, but can only move according to its underlying 6DOF. Manifolds' structures are non-Euclidean and hence prevents the direct use of classical addition and subtraction operations. However, smooth manifolds can be considered as being *locally* flat and their associated Lie algebra defines an Euclidean tangent space. $\mathbb{SE}(3)$ and $\mathbb{SO}(3)$ are both smooth manifolds with such tangent spaces defined by their associated Lie algebra $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$.

Using the mapping functions between the Lie groups and their associated Lie algebra, defined in section 2.7, one can express the pose of the camera either on its manifold space or on its Euclidean tangent space.

The idea of the *lift-solve-retract* scheme is to take advantage of these mapping operators to define the cost function in the Euclidean tangent space and to perform the states updates directly on the manifold.

The Bundle Adjustment cost function defined by Eq.(4.13) can therefore be extended to an on-manifold representation:

$$\chi^* = \arg\min_{\chi} \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \left\| \mathbf{x}_{ij} \boxminus \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) \right\|^2_{\mathbf{\Sigma}_{visual}} \tag{4.14}$$

which can be turned into the following expression:

$$\delta\chi^* = \arg\min_{\delta\chi} \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \left\| \mathbf{x}_{ij} \boxminus \pi(\mathbf{X}_i \boxplus \delta\mathbf{X}_i, {}_w\mathbf{l}_j \boxplus \delta\mathbf{l}_j) \right\|^2_{\mathbf{\Sigma}_{visual}} \tag{4.15}$$

where $\delta\chi$ defines the updates to apply to the state vector χ from its tangent space parametrization:

$$\chi^* = \chi \boxplus \delta\chi^* \tag{4.16}$$

In Eq.(4.14-4.16), the box-minus operator is the classical subtraction operation, as the visual reprojection errors are defined in $\mathbb{R}^2$, and the box-plus operator is the classical vector addition for the 3D landmarks and is defined as follows for the pose of the camera:

$$\mathbf{X}_i \boxplus \delta\mathbf{X}_i = \mathbf{X}_i \cdot \mathrm{Exp}(\delta\mathbf{X}_i) \tag{4.17}$$

where $\mathrm{Exp} : \mathbb{R}^6 \mapsto \mathbb{SE}(3)$ and $\delta\mathbf{X}_i = \begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\rho} \end{bmatrix}^T$ where $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\boldsymbol{\rho} \in \mathbb{R}^3$ respectively represent the rotational and translational coefficients of the underlying Lie algebra (see sec. 2.7). The update of $\mathbf{X}_i$ is finally computed as:

$$\mathbf{X}_i \boxplus \delta\mathbf{X}_i = \begin{bmatrix} \mathbf{R}_i \cdot \mathrm{Exp}(\delta\mathbf{X}_i(\boldsymbol{\omega})) & \mathbf{R} \cdot \mathbf{V}(\delta\mathbf{X}_i(\boldsymbol{\omega})) \cdot \delta\mathbf{X}_i(\boldsymbol{\rho}) + \mathbf{t}_i \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{4.18}$$

where $\mathbf{V}(\cdot)$ is the left Jacobian of $\mathbb{SO}(3)$.

When using a Gauss-Newton optimizer, the optimization is going to be performed as follows.

First, the cost function is lifted to the Lie algebra subspace through a first-order Taylor expansion:

$$\pi(\mathbf{X}_i \boxplus \delta\mathbf{X}_i, {}_w\mathbf{l}_j \boxplus \delta\mathbf{l}_j) \approx \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) + \mathbf{J}_{\delta\mathbf{X}_i,\delta\mathbf{l}_j}(\mathbf{X}_i, {}_w\mathbf{l}_j) \cdot \begin{bmatrix} \delta\mathbf{X}_i \\ \delta\mathbf{l}_j \end{bmatrix} \tag{4.19}$$

$$\delta\chi = \underset{\delta\chi}{\arg\min} \sum_{i\in\mathfrak{K}} \sum_{j\in\mathfrak{L}_i} \left\| \mathbf{x}_{ij} - \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) - \mathbf{J}_{\delta\mathbf{X}_i,\delta\mathbf{l}_j}(\mathbf{X}_i, {}_w\mathbf{l}_j) \cdot \begin{bmatrix} \delta\mathbf{X}_i \\ \delta\mathbf{l}_j \end{bmatrix} \right\|^2_{\boldsymbol{\Sigma}_{visual}} \tag{4.20}$$

where $\mathbf{J}_{\delta\mathbf{X}_i,\delta\mathbf{l}_j}$ is the Jacobian of the projection function $\pi(\cdot)$ over the manifold tangent space. Written in the equivalent vector form, we have:

$$\delta\chi = \underset{\delta\chi}{\arg\min} \left\| \mathbf{e}_\chi - \mathbf{J}_{\delta\chi}(\chi) \cdot \delta\chi \right\|^2_{\boldsymbol{\Sigma}_{visual}} \tag{4.21}$$

where $\mathbf{e}_\chi$ is the residuals vector.

The optimization step can then solved by taking the weighted normal equations of the problem:

$$\left(\mathbf{J}^T_{\delta\chi}(\chi) \cdot \boldsymbol{\Sigma}^{-1}_{visual} \cdot \mathbf{J}_{\delta\chi}(\chi)\right) \delta\chi = -\mathbf{J}^T_{\delta\chi}(\chi) \cdot \boldsymbol{\Sigma}^{-1}_{visual} \cdot e(\chi) \tag{4.22}$$

However, as the initial estimates of recent states can be quite far from their true values, we prefer using the Levenberg-Marquardt method:

$$\left(\underbrace{\mathbf{J}_{\delta\chi}^{T}(\chi) \cdot \mathbf{\Sigma}_{visual}^{-1} \cdot \mathbf{J}_{\delta\chi}(\chi)}_{\mathbf{A}} + \lambda \cdot \text{diag}(\mathbf{A}) \right) \delta\chi = -\mathbf{J}_{\delta\chi}^{T}(\chi) \cdot \mathbf{\Sigma}_{visual}^{-1} \cdot e(\chi) \quad (4.23)$$

where $\mathbf{A}$ is the Gauss-Newton approximation of the Hessian matrix.

After each optimization iteration, the states are updated in the *retract* step:

$$\mathbf{X}_i \leftarrow \mathbf{X}_i \cdot \text{Exp}(\delta\mathbf{X}_i) \quad ; \quad {}_{w}\mathbf{l}_j \leftarrow {}_{w}\mathbf{l}_j + \delta\mathbf{l}_j \quad (4.24)$$

In practice, because of potential numerical approximation errors, it might be required to normalize the rotational component in order to ensure that it keeps being an element of $\mathbb{SO}(3)$.

The *lift-solve-retract* scheme is hence an elegant way to optimize states defined on a manifold while keeping a euclidean structure in the real optimization process by projecting the cost function over the tangent space of this manifold.

4.6.3 Adaptive Windowed Bundle Adjustment

An ideal VSLAM algorithm would optimize a state vector χ composed of every camera poses and all the estimated 3D landmarks. However, this rapidly becomes intractable if one seeks real-time performance because of the unbounded growing of states to include in the optimization. The idea of selecting keyframes within the whole set of processed frames is a first step to reduce the number of states to optimize. Neverless, this does not bound the complexity of the Bundle Adjustment as $6 + 3N$ new parameters are going to be added to χ at each new selected keyframe (N being the number of landmarks associated to this new keyframe).

In order to prevent the unbounded growth of new states to be optimized, the Bundle Adjustment optimization is instead performed over a window of the most recent keyframes along with the 3D landmarks associated to them. In this form, it can be seen as a fixed-lag smoothing estimator.

The use of a window ensures an approximatively constant complexity when solving the NLLS problem. Moreover, when exploring an unknown environment, many keyframes will be associated to landmarks far from the

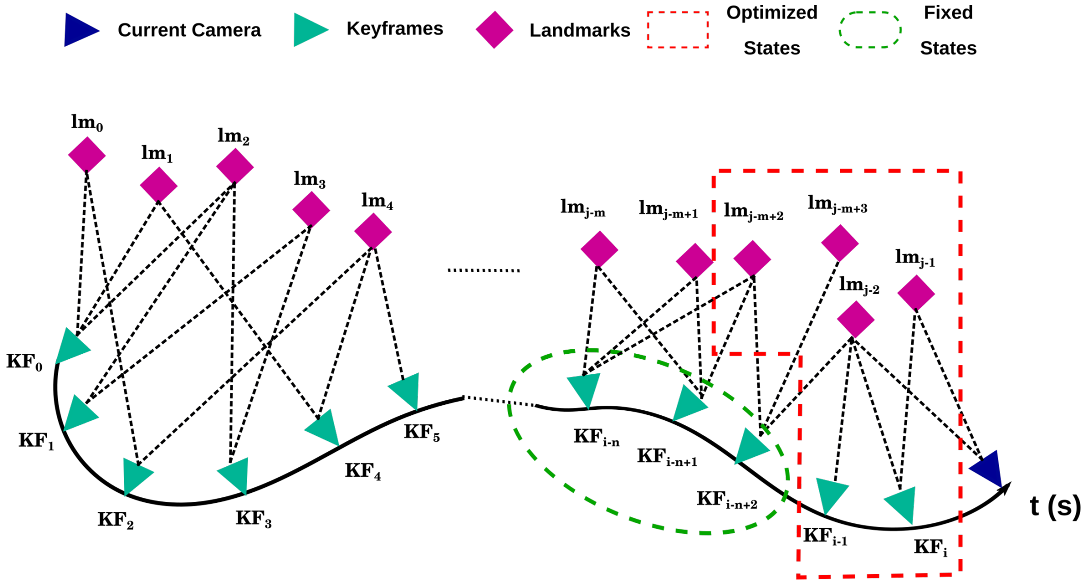


FIGURE 4.5: Illustration of the Windowed Local Bundle Adjustment. Each time a new keyframe is added, the N most recent keyframes are optimized along with the 3D landmarks they observe. Every older keyframe observing a 3D landmarks that is optimized is added to the Bundle Adjustment as a fixed constraint. As the current camera is also observing the optimized 3D landmarks, the pose estimation at frame-rate will benefit from their optimization.

current camera's position and hence have a very limited impact on the optimization of the most recent keyframes. Including older keyframes in this case would mainly act on the global consistency of the optimization results, but will have very low local effect on the most recent part of the trajectory.

On the other hand, a specific weakness of monocular VSLAM system is that the global scale is not observable and might hence drift over long trajectories. More exactly, in the monocular case, the cost function defined for the Bundle Adjustment is invariant to global change in translation, rotation and scale (7DOF). This issue is known as the problem of gauge freedom [216]. In order to fix this, two keyframes can be fixed during the optimization. Doing so, the invariance of the cost function is removed. However, in monocular VSLAM this is usually not sufficient to prevent scale drift.

One way of reducing this drift is to link current states to the oldest ones to propagate the scale of the oldest part of the trajectory to the current one. In order to limit this scale drift, we further add every keyframes associated to a 3D landmark included in the states to optimize, as fixed constraints, within the Bundle Adjustment problem. This way, many visual measurements are added to the problem without increasing the size of the state vector. The fixed keyframes will have a conditioning effect on the Bundle Adjustment outputs, as they will enforce the convergence of the optimization towards a solution compliant with their states. The number of keyframes added to the Bundle Adjustment problem is therefore set adaptively, as it depends on the number of keyframes associated to the most recent landmarks. This adaptive windowed local Bundle Adjustment is graphically presented in Fig. 4.5.

4.6.3.1 Applying a Robust Cost Function

Another challenging issue to deal with in such NLLS optimization is wrong data associations. Indeed, wrong associations would lead to outliers in the measurements that would completely violate the gaussian noise model assumptions required by the NLLS theory. The presence of even a few outliers can lead to catastrophic results, as the optimization might focus on minimizing the huge residuals induced by these outliers and return completely wrong state updates. This outliers sensitivity is due to the squaring of the residuals, which leads to very high residuals. One way to mitigate the effects of outliers within the provided measurements is to rely on a robust M-Estimator [105] to weight the residuals. Instead of applying a ℓ_2 cost to every measurement errors, robust estimators decrease the influence of measurements with too high residual. Examples of such M-Estimator are the Huber and Cauchy robust norms.

The Huber robust norm of a residual x is defined as:

$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq k \\ k \cdot \left(|x| - \frac{k}{2} \right) & \text{else} \end{cases} \tag{4.25}$$

In comparison, the effect of the Cauchy robust norm is the following:

$$\rho(x) = \frac{k^2}{2} \cdot \log \left(1 + \frac{x^2}{k^2} \right) \tag{4.26}$$

The robust norm is hence preserving the NLLS problem from a quadratic growth of the residuals and, consequently, decreases the impact of outliers.

Even if the front-end of the VSLAM algorithm is in charge of removing outliers, as explained in sections 4.5.3 and 4.5.5, it is impossible to fully ensure that no outlier remains. Therefore, we employ the Huber norm over visual measurements to make more robust the solving of the Bundle Adjustment:

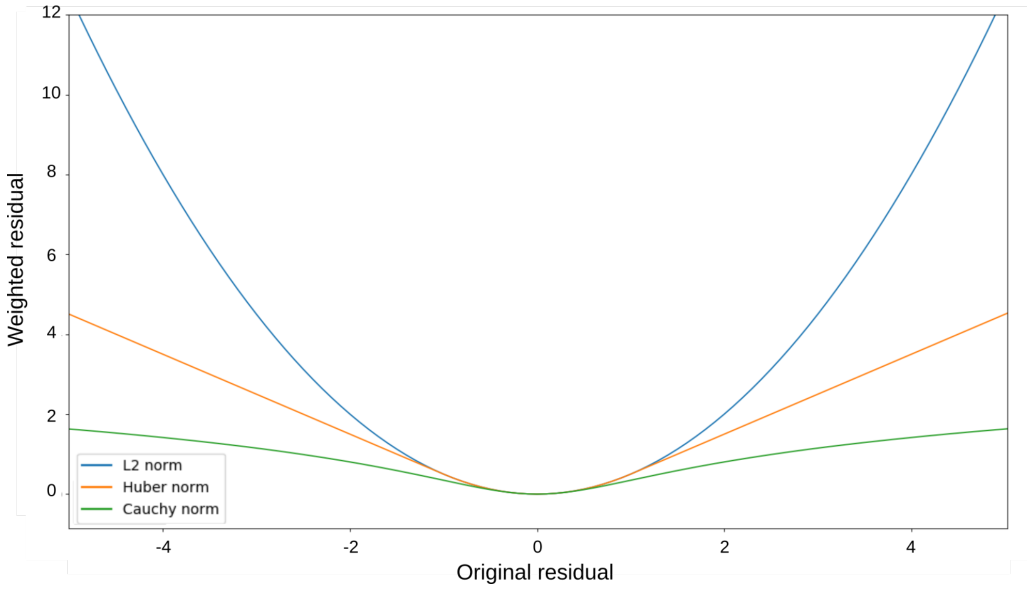


FIGURE 4.6: Robust Huber and Cauchy Norm versus ℓ_2-norm for a threshold $k = 1$.

$$\chi^* = \underset{\chi}{\arg\min} \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \left\| \mathbf{x}_{ij} - \pi(\mathbf{X}_i, {}_w\mathbf{l}_j) \right\|^{\boldsymbol{Huber}}_{\boldsymbol{\Sigma}_{visual}} \tag{4.27}$$

In practice, this new cost function is optimized by the Iteratively Reweighted Least Squares (IRLS) method:

$$\chi^* = \underset{\chi}{\arg\min} \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} w\left(\sqrt{\left\|\mathbf{e}_{ij}\right\|^2_{\boldsymbol{\Sigma}_{visual}}}\right) \cdot \left\|\mathbf{e}_{ij}\right\|^2_{\boldsymbol{\Sigma}_{visual}} \tag{4.28}$$

where $\mathbf{e}_{ij}$ is defined in Eq.(4.7) and $w(\mathbf{e}_{ij})$ is the robust weighting to apply to the residual. This robust weighting function is derived from the chosen cost function:

$$w(x) = \frac{1}{x} \cdot \frac{\partial \rho(x)}{\partial x} \tag{4.29}$$

Finally, once the IRLS done, the final residuals can be analyzed to discover potential outliers. Indeed, as the Huber weighting limits the influence of outliers, the optimization should not update the states in a way that decreases their residuals. Consequently, measurements related to a high final residual can be considered as an outlier and removed to prevent it from wrongly influence future optimization steps. Once the outliers have been identified and removed, one can also solve again the Bundle Adjustment problem without the use of a cost function in order to get more accurate estimates.

4.6.3.2 Accelerate via the Schur Complement

A last consideration that can be taken into account in order to accelerate the solving of Bundle Adjustment, is to take advantage of the special sparse structure of the underlying problem. This structure is usually very sparse as the Jacobian elements only have non-zero values between the landmarks and their observing poses. If taken into account, this sparse structure can be leveraged to highly accelerate the computation of the optimal increment $\delta\chi$ by means of Cholesky decomposition [42]. In addition to that, the landmarks are all independents in the problem and we generally have many more landmarks than poses in the state vector χ. The resulting sparse structure of the Hessian can be defined as:

$$\mathbf{H} \cdot \delta\chi = \begin{bmatrix} \mathbf{H}_{ll} & \mathbf{H}_{lX} \\ \mathbf{H}_{lX}^T & \mathbf{H}_{XX} \end{bmatrix} \cdot \begin{bmatrix} \delta\mathbf{l} \\ \delta\mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_l \\ \mathbf{b}_X \end{bmatrix} \tag{4.30}$$

where $\mathbf{H}_{ll}$ and $\mathbf{H}_{XX}$ are blocks diagonal matrices with 3×3 blocks in $\mathbf{H}_{ll}$, representing the landmark states, and 6×6 blocks in $\mathbf{H}_{XX}$, representing the keyframe states.

The Schur Complement can then be used to solve this problem [2] by first marginalizing the landmarks states in order to estimate the poses increments $\delta\mathbf{X}$ and, then, optimize the landmarks state:

$$\begin{bmatrix} \mathbf{H}_{ll} & \mathbf{H}_{lX} \\ \mathbf{0} & \mathbf{H}_{XX} - \mathbf{H}_{lX}^T \cdot \mathbf{H}_{ll}^{-1} \cdot \mathbf{H}_{lX} \end{bmatrix} \begin{bmatrix} \delta\mathbf{l} \\ \delta\mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_l \\ \mathbf{b}_X - \mathbf{H}_{lX}^T \cdot \mathbf{H}_{ll}^{-1} \cdot \mathbf{b}_l \end{bmatrix} \tag{4.31}$$

The Schur Complement formulation hence reduces the size of the matrix to invert and takes advantage of the sparse structure of the problem.

4.6.3.3 Summary of the Bundle Adjustment Solving

We now summarize the way the Bundle Adjustment is solved in UW-VO. First, we define the states to optimize, that is the states to add to the χ state vector, as the N most recent keyframes and the M 3D landmarks observed by these keyframes. We hence use a sliding window formulation to define the states. This sliding window is further enhanced by adding, as fixed states, older keyframes associated to at least one of the M landmarks. By making the size of the sliding window adaptive, we limit the scale drift that might happen when solving the Bundle Adjustment problem. The optimization is then performed with the Levenberg-Marquardt method. The cost function to minimize is turned into the Huber form in order to be robust to potential outliers within the measurements. The optimization is hence executed as an IRLS estimation. As the keyframes states to optimize are elements of the $\mathbb{SE}(3)$ manifold, the minimization of the cost function is solved by *lifting* the problem onto the tangent space, *solving* it and applying the estimated states' updates by *retracting* these updates onto the manifold. Taking advantage of the special sparse structure of the Bundle Adjustment problem to speed-up the optimization process, the landmarks are first marginalized onto the keyframes' poses through the use of the Schur Complement. This results in a two steps optimization as the updates to apply to the poses are computed first and, then, the marginalized landmarks updates are computed. The optimization is performed with a fixed number of iterations to limit the computation time allowed to the Bundle Adjustment. Once finished, every measurements whose final residual is higher than a threshold are removed. The Bundle Adjustment is finally run once again on the inliers only, with the standard ℓ_2 norm. This last step is usually really fast because the states should already be almost optimal.

As a result, after the selection of every new keyframe, the most recent part of the trajectory gets optimized and, by optimizing the set of the most recent landmarks, we ensure that the pose estimation of the next frames will be done by considering only statistically optimal 3D landmarks.

4.7 Experimental Results

Implementation: The proposed method has been developed in C++ and uses the ROS middleware [184]. The tracking of features is done with the OpenCV implementation of the Kanade-Lucas algorithm [21]. Epipolar geometry and P3P pose estimations are computed using the OpenGV library [128]. Bundle Adjustment is performed using the graph optimization framework g2o [132] and runs in a parallel thread.

We use the following settings in UW-VO: the covariances of the visual measurements are set as the identity matrix, the Huber threshold is set to 5.991 (*i.e.* a probability of 95%) and we set the number of keyframes to optimize to 10. The average run time is of 25 ms per frame, with the tracking set to 250 features per frame. The run time goes up to 35 ms when a new keyframe is required because of the features detection and triangulation overload. The Bundle Adjustment optimization takes around 10 ms but the whole process in the back-end thread takes longer because of the use of mutex. Thus UW-VO can run in real time for video sequences with a frame rate up to 30 Hz. The experiments have been carried out with an Intel Core i5-5200 CPU-2.20GHz-8 Gb RAM.

To the best of our knowledge, there is no underwater method able to estimate localization from monocular images available open-source. Furthermore, no publicly available datasets were released with these methods, so we cannot compare our method with them. Hence, UW-VO has been evaluated and compared to three state-of-the-art open-source monocular SLAM methods: ORB-SLAM [161], LSD-SLAM [62] and SVO [81]. The evaluation has been conducted on different datasets which are all available online, allowing future methods to compare to our results.

All the algorithms are evaluated on real underwater datasets. In addition, UW-VO and ORB-SLAM are also evaluated on a simulated dataset [54], whose frame rate (10 Hz) is too low for SVO and LSD-SLAM. Indeed, SVO and LSD-SLAM are direct methods which require very high overlap between two successive images in order to work. Please note that ORB-SLAM and SVO have been fine-tuned in order to work properly. For ORB-SLAM, the features detection threshold was set at the lowest possible value and the number of points was set to 2000. For SVO, the features detection threshold was also set at the lowest possible value and the number of tracked features required for initialization was lowered to 50.

For each method, every results presented are the averaged results over five runs. Moreover, as we are comparing monocular systems here, the trajectories have to be aligned and scaled with respect to the groundtruth in order to compare them. The alignment and scaling is performed by computing a similarity transformation using the method of [217].

4.7.1 Results on a Simulated Underwater Dataset

A simulated dataset created from real underwater pictures has been made available to the community by [54]. Four monocular videos of a triangle-shaped trajectory are provided, with four different levels of turbidity. The turbidity effects are simulated by corrupting each video sequence with an certain amount of visual noise. The turbidity effect is increased between each sequence and ends up turning the original images into more blueish versions (see Fig. 4.7). Even if VSLAM systems process images in their gray-scale form, the increase in turbidity has an impact on the tracking as it reduces the contrast of the images. The images resolution of these videos is 320x240 pixels. In each sequence, the triangle-shaped trajectory is performed twice and it starts and ends at the same place. These four sequences have been used to evaluate the robustness against turbidity of UW-VO with respect to ORB-SLAM. For fair comparison, ORB-SLAM has been run with and without its loop-closing feature. We will refer to the version without loop-closure as V.O. ORB-SLAM in the following.

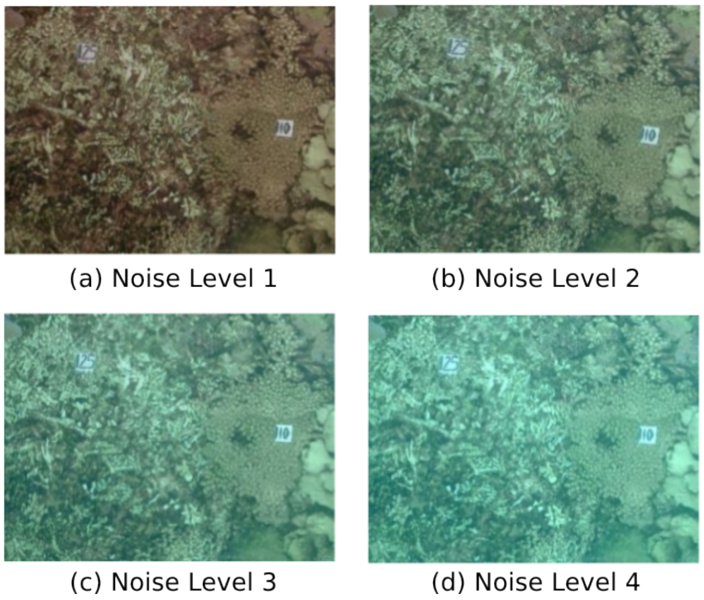


FIGURE 4.7: The four different turbidity levels of the simulated dataset [54].

Table 4.1 presents the final drift at the end of the trajectory for each method. On the first three sequences, ORB-SLAM is able to close the loops and therefore has the lowest drift values, as the detection of the loop closures allows to reduce the drift accumulated in-between. On the same sequences, V.O. ORB-SLAM has the highest level of drifts. Note that ORB-SLAM and its V.O. alternative fail half the time on the third level of noise sequence and have been run many times before getting five good trajectories. It is worth noting that the localization drift increases significantly for V.O. ORB-SLAM when the turbidity level gets higher. This is mostly due to the increased inaccuracy in its tracking of ORB features. On the last sequence, the turbidity level is such that ORB descriptors get too ambiguous and leads to failure in ORB-SLAM tracking. These results highlight the deficiency of ORB-SLAM tracking method on turbid images.

In comparison, UW-VO is able to run on all the sequences, including the ones with the highest levels of noise (Fig. 4.8). The computed trajectories are more accurate than V.O. ORB-SLAM and we can note that it is barely affected by the noise level (Fig. 4.9). These results confirm the efficiency of UW-VO as a robust VSLAM system in turbid environments.

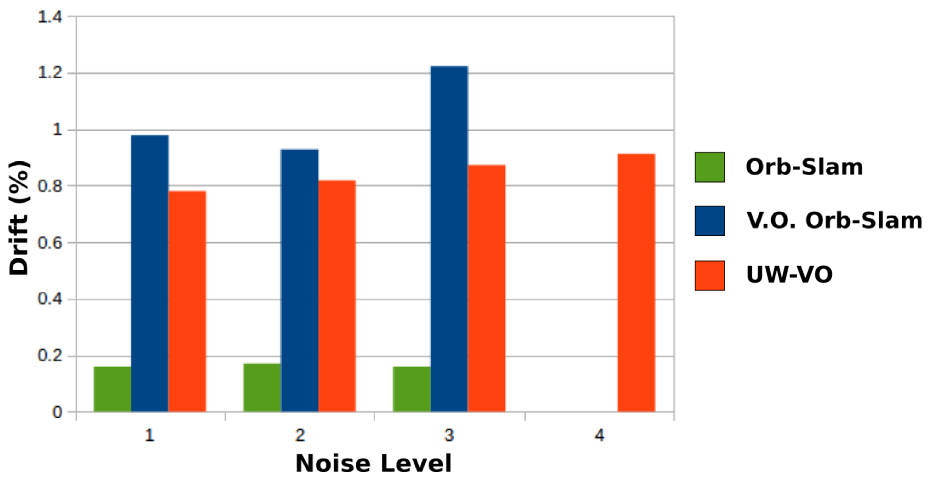


FIGURE 4.8: Drift of ORB-SLAM (green), V.O. ORB-SLAM (blue) and UW-VO (red) on the simulated underwater dataset [54].

TABLE 4.1: Translation drift (in % of the traveled distance) on the simulated underwater video sequences with different level of turbidity. The given results are the average drift over five runs for each algorithm. V.O. ORB-SLAM designates ORB-SLAM without the loop closing feature enabled. ORB-SLAM results are given for information. The (*) denotes very frequent failure of the algorithm.

		Drift (in %)		
Seq. Noise Level	Turbidity	*ORB-SLAM*	V.O. ORB-SLAM	UW-VO
1	None	*0.18*	0.97	**0.78**
2	Low	*0.18*	0.93	**0.81**
3	Medium	*0.17**	1.21*	**0.85**
4	High	*X*	X	**0.89**

4.7.2 Results on a Real Underwater Video Sequences

We now present experiments conducted on five real underwater video sequences. These sequences were gathered 500 meters deep in the Mediterranean Sea (Corsica), in 2016, during an archaeological mission conducted by the French Department of Underwater Archaeological Research (DRASSM). The videos were recorded from a camera embedded on an ROV and grayscale 640x480 images were captured at 16 Hz. The calibration of the camera

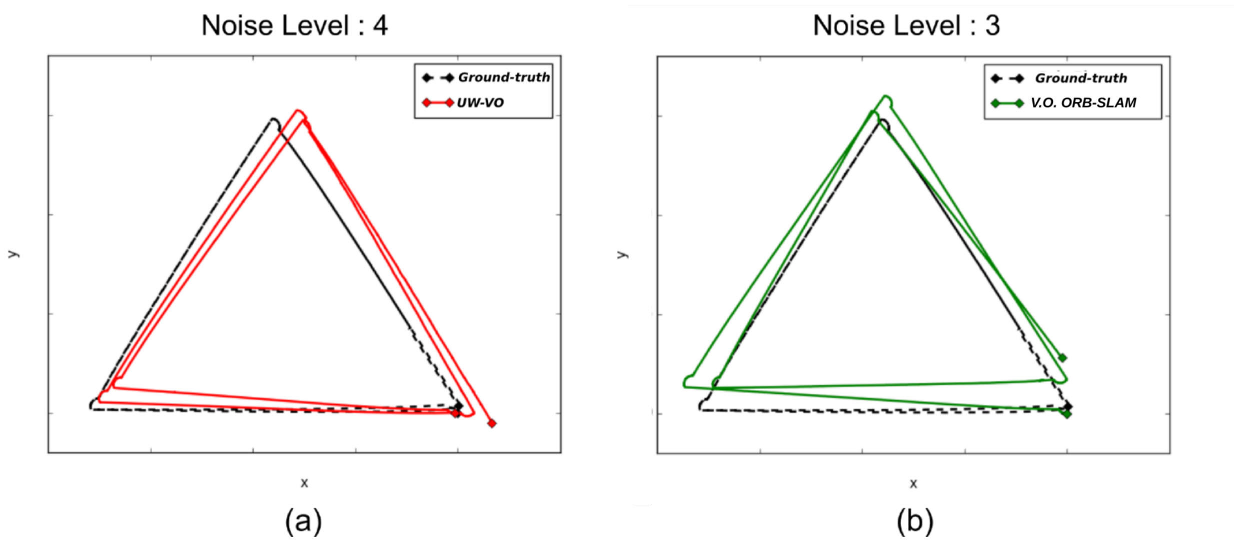


FIGURE 4.9: Trajectories estimated with (**a**) UW-VO on the sequence with a high level of turbidity and with (**b**) V.O. ORB-SLAM on the sequence with a medium level turbidity.

has been done with the Kalibr [82] library. Calibration was done *in situ* in order to estimate the intrinsic parameters and the distortion coefficients of the whole optical system. The camera recording the videos was placed inside an underwater housing equipped with a spherical dome and we obtained good results using the pinhole-radtan model (assessed by a reprojection error $<$ 0.2 pixel).
These five sequences can be classified as follows:

- Sequence #1: low level of turbidity and almost no fish.
- Sequence #2: medium level of turbidity and some fishes.
- Sequence #3: high level of turbidity and many fishes.
- Sequence #4: low level of turbidity and many fishes.
- Sequence #5: medium level of turbidity and many fishes.

For each of these sequences, a ground truth was computed using the state-of-the-art Structure-from-Motion software Colmap [197]. Colmap computes trajectories offline by exhaustively trying to match all the images of a given sequence, thus finding many loops and creating very reliable trajectories. We could assess the accuracy of the reconstructed trajectories both visually and by checking the validity of the matched features between the images.

Here, we compare ORB-SLAM, LSD-SLAM and SVO to UW-VO. We evaluate the results of each algorithm against the trajectories computed offline by

Colmap by computing the absolute trajectory error [212] (Fig. 4.10). The results are displayed in Table 4.2. To observe the effect of the retracking mechanism described in section 4.5.4, we have run the UW-VO algorithm with and without enabling this feature, respectively referring to it as UW-VO and UW-VO*.

TABLE 4.2: Absolute trajectory errors (RMSE in % of the traveled distance) for five underwater sequences with different visual disturbances. The given results are the average drift over five runs for each algorithm. UW-VO* designates our method without the retracking step, while UW-VO designates our method with the retracking step.

				Absolute Trajectory Error RMSE(in %)				
Seq. #	Duration	Turbidity Level	Short Occlusions	LSD-SLAM	ORB-SLAM	SVO	UW-VO*	UW-VO
1	4′	Low	Few	X	1.67	**1.63**	1.78	1.76
2	2′30″	Medium	Some	X	1.91	2.45	1.78	**1.73**
3	22″	High	Many	X	X	1.57	1.10	**1.04**
4	4′30″	Low	Many	X	**1.13**	X	1.61	1.58
5	3′15″	Medium	Many	X	1.94	X	2.08	**1.88**

As we can see, LSD-SLAM fails on all the sequences. This is most likely due to its semi-dense approach based on the tracking of edges with strong gradients, which are not frequent on deep-sea images. SVO computes accurate trajectories on the sequences that are barely affected by dynamism from moving fishes. The tracking of SVO, which is similar to optical flow, works well even on turbid images, but its direct pose estimation method is not robust to wrongly tracked photometric patches like the ones created by moving fishes (seq. #3, #4, #5). ORB-SLAM on the other hand performs well on highly dynamic sequences, but loses in accuracy when turbidity is present (seq. #2, #3, #5). Its pose estimation method, based on the observations of independent features, is hence robust to short occlusions and dynamic objects, but its tracking method fails on images degraded by turbidity. Furthermore, we can note that despite loop closures in the trajectories (see Fig. 4.10), ORB-SLAM is not able to detect them. The failure to detect the loop closures indicates that the Bag of Words approaches [85] might not be suited to the underwater environment, which does not provide many discriminant features.

UW-VO is the only method able to run on all the sequences. While the estimated trajectory is slightly less accurate on the easiest sequence (seq. #1), UW-VO performs better than ORB-SLAM and SVO on the hardest sequences

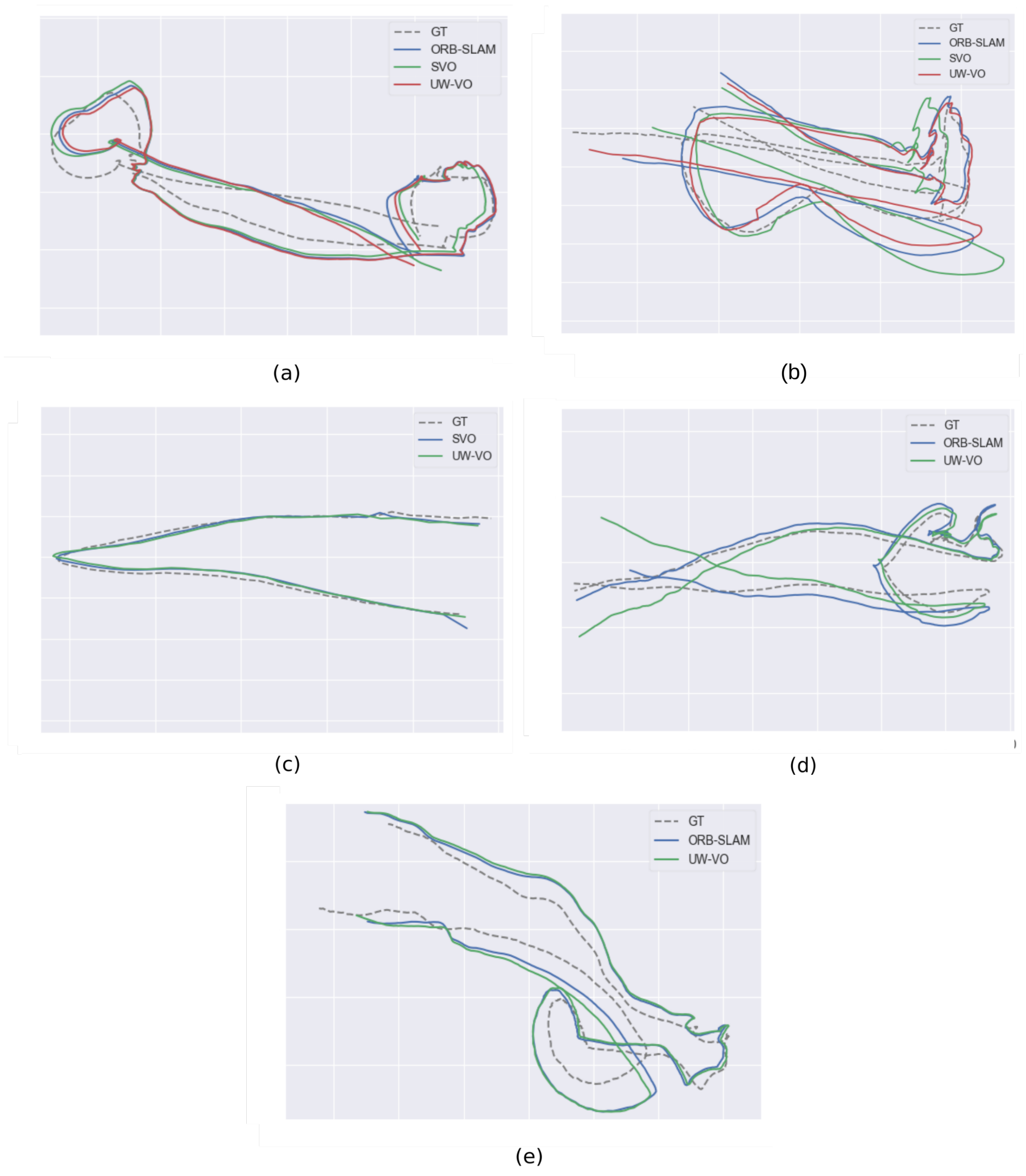


FIGURE 4.10: Trajectories of ORB-SLAM, SVO and UW-VO over the five underwater sequences. (**a**) Sequence 1, (**b**) Sequence 2, (**c**) Sequence 3, (**d**) Sequence 4, (**e**) Sequence 5. Ground-truths (GT) are extracted from Colmap trajectories.

(seq. #2, #3, #5, with turbidity and dynamism, which is very common during underwater archaeological operations). We can see the benefit of the developed retracking mechanism on most of the sequences. Nonetheless, this optical flow retracking step is not as efficient as the use of descriptors when the number of short occlusions is very large (seq. #4). Studying the effect of combining optical flow tracking with the use of descriptors could result in an interesting hybrid method for future work.

4.8 Conclusion

In this chapter we have presented UW-VO, a new vision-based underwater localization method. We propose a keyframe-based monocular VSLAM method, robust to typical disturbances of the underwater environment. We use the KLT method to track features frame to frame through optical flow. We further enhanced the KLT by adding a retracking mechanism, making it robust to short occlusions due to the environment dynamics. We have shown that UW-VO outperforms the state-of-the-art VSLAM algorithms ORB-SLAM, LSD-SLAM and SVO in terms of robustness on underwater video sequences.

Some perspectives would be to combine the use of the KLT with descriptors as a way of retracking lost features on a larger scale. Furthermore, these descriptors could be used for the purpose of loop-closure detection. However, we have observed that loop-closing approaches based on classical Bag of Words [85] do not work as expected in our tests (see the results obtained with ORB-SLAM) and alternative methods, more suited to underwater images, in the lead of [29, 30] need to be investigated. Also, online Bag of Words methods [5, 168, 93] seems promising for this task as they are incrementally built online and are therefore always adapted to the current environment.

As any monocular VSLAM method, UW-VO do not estimate scaled trajectories and scale drift is unavoidable on long trajectories. The problem of scale drift can be tackled by the detection of loop-closures. However, this is only efficient when the robot is often coming into previously mapped areas. A way for both recovering the scale of the trajectories and to limit drift is to fuse additional sensors within a monocular VSLAM. This will be the purpose of the next chapter in which will present an extension of UW-VO that integrates the measurements of a pressure sensor and a low-cost MEMS-IMU.

Chapter 5

Tight Visual-Inertial-Pressure Fusion for Underwater Localization

Contents

5.1 Introduction

In the previous chapter, we have proposed UW-VO, a monocular Visual SLAM algorithm, robust to the visual disturbances produced by underwater environments. However, as any pure monocular localization method, the metric scale of the trajectories estimated by UW-VO is unknown and scale drift is unavoidable on long trajectories.

This chapter presents a localization algorithm based on the fusion of a monocular camera with a low-cost *Micro Electro-Mechanical System Inertial Measurement Unit* (MEMS-IMU) and a pressure sensor. The goal of the fusion process presented in this chapter is both to recover the scale of the trajectories and to improve the localization estimations. Moreover, adding an IMU and a pressure sensor provides motion information that can be useful during short period of poor visual information.

The contributions of this chapter are the following:

- A deep investigation about the fusion of a pressure sensor with a monocular camera for the purpose of localization.
- A tight fusion of vision and pressure to perform Visual-Pressure SLAM.
- A Visual-Inertial-Pressure SLAM extension that further integrates an IMU thanks to the use of preintegration.

This chapter is organized as follows. We first review the fusion techniques that have been applied in previous works when dealing with vision-based setups. Then, we investigate the fusion of vision and pressure and propose different strategies to solve the localization problem with such a setup. Finally, we present an extension of this Visual-Pressure SLAM by leveraging preintegrated IMU measurements. Both the Visual-Pressure and Visual-Inertial-Pressure SLAM methods are evaluated on real data sequences extracted from the AQUALOC dataset [72]. This dataset will be presented in the next chapter.

5.2 Related Works

The fusion of IMU's measurements with monocular cameras has received quite a lot of attention from the community because of its ability in retrieving the scale factor and handling fast motions.

A first set of works tackled this problem by loosely coupling the visual and IMU sensors. In loosely coupled systems, visual and inertial measurements are individually processed and their individual pose estimations are then fused together (usually with an EKF). This fusion paradigm has been used in [96, 221] to feed an EKF with the measurements of an IMU and speed terms derived from the pose estimated from vision only. A modular framework for loosely coupling additional sensors, not just IMUs and cameras, was also proposed in [149].

The weakness of loose fusion is that complementary measurements are not tied together for the purpose of state estimation. To cope with this, tight fusion estimators have been proposed in the literature. When used to couple cameras and IMUs, the visual and inertial measurements are directly processed altogether. A direct benefit is that more data are used for pose estimation than in loosely coupled approaches, where the visual measurements are converted into a single pose estimation. Furthermore, the tight coupling is very useful when using visual measurements as it allows to early detect potential outliers. However, they come at the price of a higher computational complexity compared to loose approaches. They also usually require a finer modeling of the sensors than loose approaches because of the tight fusion of the different sensors measurements in the optimization steps.

Tight fusion estimators have been proposed with filtering paradigm in [143, 142, 160, 20, 44] and optimization frameworks [137, 138, 183, 162, 81]. The optimization frameworks can be seen as enhancement of the Bundle Adjustment operations in VSLAM by adding IMU related measurements into it. They usually provide more accurate estimations than filtered-based methods but at the expense of more computation [138]. Some works also used to tight fusion estimators with additional sensors such as odometers [226] or array of magnetometers [36, 35].

In underwater robotics, the filtering method of [160] has been extended in [201] to include a pressure sensor which provides measurement updates applied to constrain the estimated position along the z-axis. More recently, an extension of [138] was proposed in [185] to include the measurements of a profiling sonar. A very light framework was also proposed in [47] to iteratively estimate the position of a ROV from a monocular camera whose measurements are scaled and corrected by means of a low-cost IMU and a pressure sensor. However this method is mostly restricted to horizontal servoing in small areas.

With respect to these works, we follow the idea of tightly coupling different sensors. However, to the best of our knowledge, there is no previous work about tightly coupling a monocular camera, a pressure sensor and a low-cost IMU.

5.3 Visual-Pressure SLAM

We first investigate the fusion of depth measurements from a pressure sensor within UW-VO. By adding depth measurements to the monocular system, we seek the recovery of the metric scale of the estimated trajectories, scale drift elimination and global improvements in the localization estimations.

5.3.1 Pressure Sensor

Pressure sensors measure the applied force per unit of surface area. When dived into a fluid on Earth, a pressure sensor senses the amount of pressure due to the fluid in addition to the pressure due to the atmosphere. The pressure due to a non-compressible fluid is directly proportional to the fluid level, that is to the height of the fluid above the pressure sensor. The pressure measurements are related to this height through the following equation:

$$\tilde{P} = \rho_{\text{fluid}} \cdot g \cdot d + P_{atm} \tag{5.1}$$

where $\tilde{P}$ and P_{atm} are respectively the measured pressure and the atmospheric pressure (in Pa), ρ_{fluid} is the fluid density (in kg.m^{-3}), g is the gravitational constant ($\approx$ 9.81 m.s^{-2}) and d is the height of the fluid column above the sensor (in meters).

The height of the fluid column corresponds to the actual depth of the sensor with respect to the surface. This depth is hence directly related to the pressure measurements:

$$\tilde{d} = \frac{\tilde{P} - P_{atm}}{\rho_{\text{fluid}} \cdot g} \tag{5.2}$$

The use of a pressure sensor for underwater localization hence provides an absolute, one-dimensional, position information.

In our SLAM problem, we do not consider global positioning but local positioning, that is, the trajectory performed by the robot from the initial position at $t = 0$. Hence, instead of using the raw depth measurements in the algorithm, we use the depth difference between the current measurement and the depth measured at the initialization time. In what follows, the depth values will always correspond to:

$$\tilde{d}_i = {}_{\text{raw}}\tilde{d}_i - {}_{\text{raw}}\tilde{d}_0 \tag{5.3}$$

where $\tilde{d}_i$ is the depth value linked to the state at time i, ${}_{\text{raw}}\tilde{d}_i$ is the direct depth measurement computed from the pressure received at time i, and ${}_{\text{raw}}\tilde{d}_0$ is the first raw depth measured at time 0, that is at the initialization time. All the depth measurements are expressed in meters.

Besides, in our case, we decided to model the depth measurements as corrupted by gaussian noise:

$$\tilde{d}_i = \overline{d}_i + \eta_{depth} \quad , \quad \eta_{depth} \sim \mathcal{N}\left(0, \sigma^2_{depth}\right) \tag{5.4}$$

where $\overline{d}_i$ is the true depth and η_{depth} is the zero-centered depth gaussian noise of standard deviation σ_{depth}.

5.3.2 Problem Statement

We first recall the states of interest of our SLAM problem. We use the same notations that in the previous chapter and define the state of interest as:

$$\chi_i = \begin{bmatrix} \mathbf{X}_{wc_i} & \mathbf{X}_{wk} & {}_w\mathbf{l}_m \end{bmatrix}^T \tag{5.5}$$

where $\mathbf{X}_{wi}$ is the pose of the current frame, $\mathbf{X}_{wk}$ is the set of previously selected keyframes and ${}_w\mathbf{l}_m$ the set of estimated 3D landmarks.

In this section, we propose to tightly fuse vision and pressure measurements within our SLAM system. In opposition with loose fusion methods, the tight fusion of vision and pressure allows the visual part of the algorithm to benefit from the pressure measurements. This is especially useful to early detect potential outliers within the visual measurements. In order to follow the factor graph formulation of the Visual SLAM, we formulate the fusion by adding depth measurements as new factors linked to the camera pose states.

While the fusion of depth measurements might seem trivial given their simplicity, correctly coupling them within a monocular VSLAM algorithm is actually tricky and many considerations have to be taken into account. First, as the monocular setup cannot recover the metric scale of its estimations, the depth measurements have to be used to recover the visual scale factor. Secondly, there is a reference frame issue (see Fig. 5.1). Indeed, depth

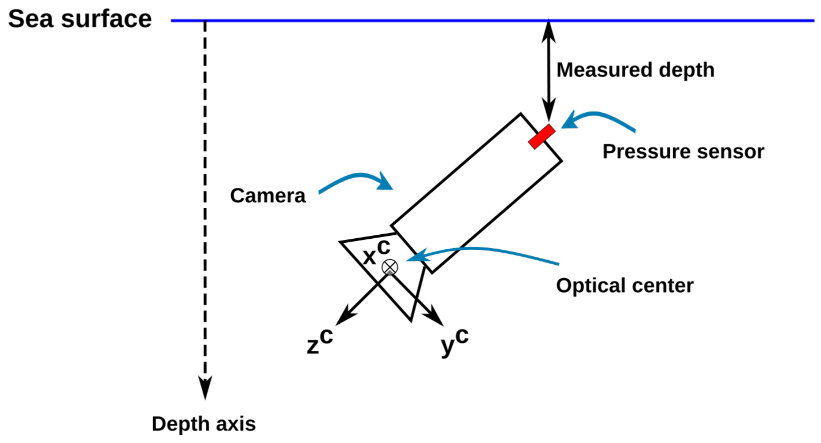


FIGURE 5.1: Camera - Pressure sensor setup. The pressure sensor provides depth measurements which are given along the depth axis, orthogonal to the surface. The camera's frame $(\vec{x^c}, \vec{y^c}, \vec{z^c})$ is arbitrarily defined by its physical orientation and the position of its optical center.

measurements are given along a specific axis, that can be considered as orthogonal to the water surface and aligned with the gravity axis. If we use a frame of reference that is aligned with this axis, we can directly relate the depth measurements to the robot's positions. For instance, if an accelerometer is available, gravity can be observed and the frame of reference for the pose estimations can be aligned with this axis. Furthermore, if a compass or a fiber optic gyroscope (FOG) is also available, the frame of reference can be set in a NED (North-East-Down) or ENU (East-North-Up) configuration. However, with a monocular camera there is no way of measuring any of this. In pure VSLAM, the frame of reference is arbitrary and is usually defined as being aligned with the first camera's pose. In UW-VO, the first keyframe is used as the reference frame and its pose is represented by a zero vector for its position and an identity matrix for its rotation matrix.

5.3.2.1 Camera-Depth misalignment Effect

The pose estimations performed by a monocular VSLAM are always done relatively to a frame of reference C_0 (defined as the first selected keyframe in our case). The physical installation of the camera on the robot mostly determine the orientation of C_0 with respect to the depth axis. As visual pose estimations are always performed relatively to a 3D map whose landmarks are defined in C_0 (fixed at initialization by the first 3D mapping), the randomness that exists on C_0 does not impact at all VSLAM algorithms. However, when fusing a monocular camera with a pressure sensor, this turns out to

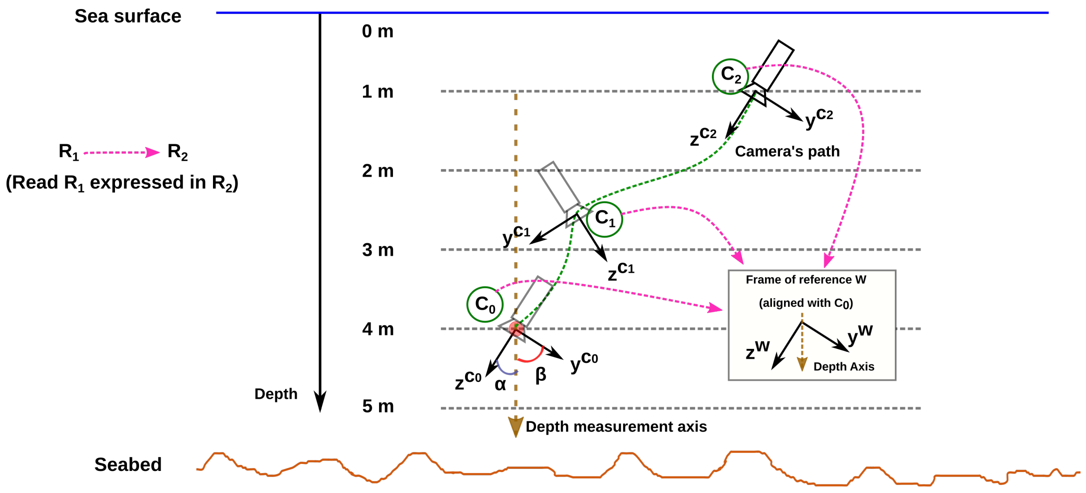


FIGURE 5.2: Difference between depth measurements and the frame of reference of camera's poses. C_0, C_1 and C_2 represent the camera's frame at time 0, 1 and 2. C_0 is the initial frame and the following poses are expressed with respect to this frame. However, this frame of reference is not aligned with the depth axis.

be a real issue. On the one hand, depth measurements are given along a specific axis and, on the other hand, we have a VSLAM method that cannot relate the reference of its estimation to any real-world physical constant. In the underwater context, bottom looking camera are very often used. By bottom looking we mean that the field of view of the camera is mostly oriented towards the seabed. Nevertheless, as we have no way of ensuring that the camera's z-axis will be perfectly aligned with the depth axis, we can assume that there will always be some misalignment between the depth axis and the (non-orthogonal) axes of the frame of reference C_0. Therefore, if one wants to integrate depth measurements to constrain the visual pose estimations, the depth measurement should be projected into C_0 in order to recover its implication along the axes of C_0. Yet, with no way of knowing the misalignment between C_0 and the depth axis, this is not possible.

If we assume a bottom looking camera, we can further assume that the majority of the depth measurements contribute to constraining the estimation of the position along the z-axis of C_0. However, as illustrated in Fig. 5.2, the existing misalignment will lead to the following relation between the depth measurements and the pose estimated by the VSLAM algorithm along the z-axis:

$$\overline{d}_i = \overline{t^z}_{c_0 c_i} \cdot \cos(\alpha) \tag{5.6}$$

where $\overline{d}_i$ is an ideal noiseless depth measurement at frame i, $\overline{t^z}_{c_0 c_i}$ is the true robot's position along the z-axis in C_0 at frame i and α is the misalignment angle that exists between the depth axis and the z-axis of C_0.

We can see that the error between the depth measurements and the true robot position is linear in $\cos(\alpha)$. The bigger the depth variations are, the bigger the error is. Furthermore, the magnitude of $\cos(\alpha)$ will highly impact the growth of the error rate.

The misalignment error in depth can be seen as a varying error depending on an unknown parameter, the α angle. Properly handling this issue would require to adjust the confidence given to the depth measurements which is quite difficult without knowing the α angle.

In the following, we will drop the C_0 notation and use instead w to denote the frame of reference. This is to stay consistent with the notations in the previous chapters but also to be more general as defining C_0 as the frame of reference is purely arbitrary.

5.3.2.2 Practical Assumptions

We make some assumptions about the visual-pressure acquisition system. First, we consider a bottom looking camera, that is a camera oriented toward the seabed, but without assuming a purely vertical system. This allows the development of a localization solution flexible enough to work with different setups. In practice, embedding a camera on an underwater robot might be constrained by the other embedded equipments, which can prevent setting up the camera vertically.

Secondly, we consider a system equipped with a camera and a pressure sensor rigidly mounted. Last, we consider two kinds of pressure sensors. A first kind with an acquisition rate lower than the camera's one and another kind with a higher acquisition rate. In the first case, we face a system which will have no pressure measurements linked to some frames. In the second case, we are ensured to receive several pressure measurements between every acquired frame. Once again, this assumption is made to fit different practical cases that can be met in the context of underwater operations. For both kinds of pressure sensors, we further approximate that, for a camera running at 20 Hz and given the low dynamics of ROVs, a pressure measurement received at time t, between a frame taken at t_i and another at t_j (with $t_j > t_i$), the time difference $\delta t = t_j - t$ is not significant in front of the pressure sensor accuracy. Hence, in the following, we consider any pressure measurement taken right before the acquisition of a new frame as measuring the depth of this frame at its acquisition time. If several pressure measurements are received between two consecutive frames i and j, we average the M measurements received to create a single measurement:

$$\tilde{d}_j = \frac{1}{M} \cdot \sum_{k=0}^{M-1} \tilde{d}_k \tag{5.7}$$

In this case, the measurement noise is updated as $\hat{\sigma}_{depth} = \sigma_{depth} / \sqrt{M}$. For notational convenience, we will simply refer to $\hat{\sigma}_{depth}$ by σ_{depth} in what follows. This way, the notations will suit both kinds of pressure sensors.

5.3.3 Depth Integration Strategies

We next describe the developed strategies to tackle this visual-pressure localization problem. We first explore different ways of integrating the pressure measurements within the nonlinear least-squares (NLLS) optimization of the Visual SLAM algorithm. Then, we propose an initialization procedure to handle the initialization of scale and the camera-depth misalignment issue.

For all the strategies, we propose a first modification of the SLAM algorithm which ensures having a depth measurement linked to every selected keyframe. This is done by conditioning the keyframe creation process to the presence of a depth measurement in addition to the visual tracking quality thresholds (see 4.5.6 page 78).

5.3.3.1 Strategy 1: Integration of absolute depth measurements

The first fusion strategy explored consists in directly integrating the depth measurements as absolute constraints. This strategy tackles the fusion problem by adding a depth error term directly on their associated pose during the pose estimations. This error term is defined as the following Mahalanobis distance:

$$E_{depth}(\mathbf{X}_i) = \| \tilde{d}_i - \hat{t^z}_{Wc_i} \|^2_{\sigma^2_{depth}} \tag{5.8}$$

where $\hat{t^z}_{Wc_i} \in \mathbb{R}$ is the z-axis component of the translational part of the pose $\mathbf{X}_i$.

The depth related error terms are then used both for the pose estimations at frame rate and for the Bundle Adjustment part of the VSLAM algorithm. The estimation of pose at frame-rate is the one done in the front-end thread

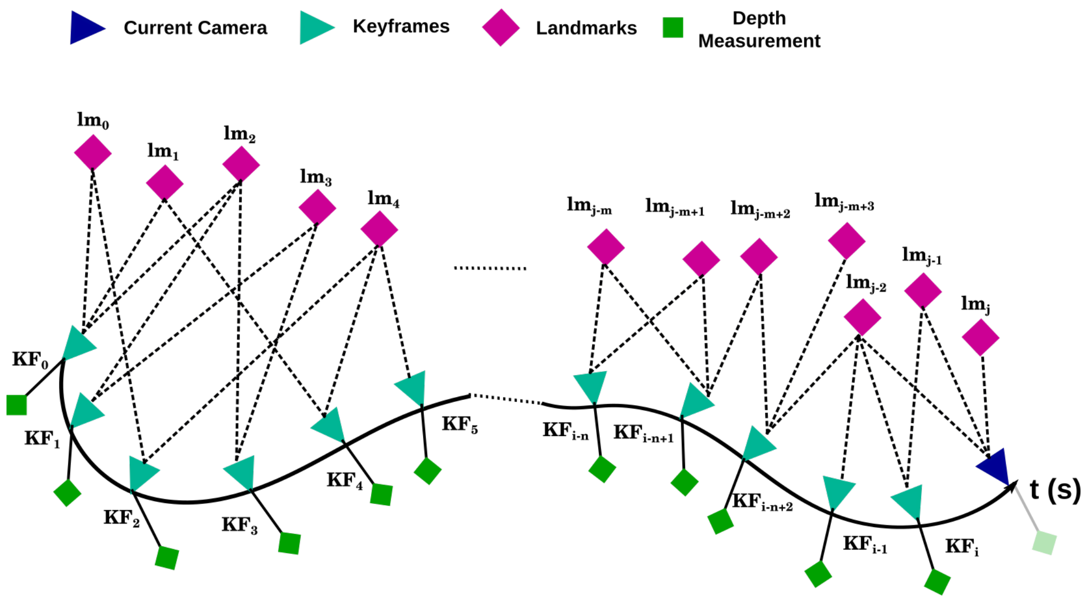


FIGURE 5.3: Strategy 1: Visual-Pressure SLAM with absolute depth measurements. This schema represents the factor graph that can be derived from this SLAM problem. The states to optimize are the poses of the keyframes and of the current frame along with the visual landmarks. The different states are bound together by visual measurements (dark dotted line) and absolute depth measurements are linked to each pose (green squares).

of UW-VO (see section 4.5) and the Bundle Adjustment is the optimization operation performed in the back-end thread of UW-VO (see section 4.6).

Frame-rate Pose Estimation

For the pose estimation at frame rate, we first estimate the current pose from vision-only measurements with the P3P method described in section 4.5.5 (page 77). Then, if a depth measurement is linked to the current frame, we include it in the pose refinement step. So, instead of optimizing the pose estimate with visual measurements only, we add the depth error term to the cost function:

$$\mathbf{X}_i^* = \underset{\mathbf{X}_i}{\arg\min} \left(E_{visual}\left(\mathbf{X}_i\right) + \beta \cdot E_{depth}\left(\mathbf{X}_i\right) \right) \tag{5.9}$$

with $\beta = 0$ if no depth measurements are linked to frame i or $\beta = 1$ otherwise, and $E_{visual}\left(\mathbf{X}_i\right)$ defined as:

$$E_{visual}\left(\mathbf{X}_i\right) = \sum_{j \in \mathfrak{L}_i} \rho \left(\mathbf{e}_{ij}^T \cdot \mathbf{\Sigma}_{visual}^{-1} \cdot \mathbf{e}_{ij} \right) \tag{5.10}$$

where $\mathfrak{L}_i$ is the set of 3D landmarks observed by frame i and $\rho(\cdot)$ is the robust Huber cost function. The error terms $\mathbf{e}_{ij}$ are the reprojection errors, as defined in section 4.2 (page 70).

Depth-enhanced Bundle Adjustment

Equation 5.10 allows to benefit from depth measurements in the pose estimation at frame-rate. We can further extend the Bundle Adjustment to benefit from these measurements as well. We do it by extending the factor graph defining the Bundle Adjustment step of UW-VO as illustrated in Fig. 5.3. The resulting optimization is therefore modeled as:

$$\chi^* = \underset{\mathbf{X}_i}{\arg\min} \left(E_{visual}(\chi) + E_{depth}(\chi) \right) \tag{5.11}$$

where $E_{visual}(\chi)$ and $E_{depth}(\chi)$ are defined as:

$$E_{visual}(\chi) = \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \rho \left(\mathbf{e}_{ij}^T \cdot \mathbf{\Sigma}_{visual}^{-1} \cdot \mathbf{e}_{ij} \right) \tag{5.12}$$

$$E_{depth}(\chi) = \sum_{i \in \mathfrak{K}} E_{depth}(\mathbf{X}_i) = \sum_{i \in \mathfrak{K}} \| \tilde{d}_i - \hat{t}^z{}_{Wc_i} \|^2_{\sigma^2_{depth}} \tag{5.13}$$

with $\mathfrak{K}$ the set of keyframes included in χ and $\mathfrak{L}_i$ the sets of landmarks observed by these keyframes.

This strategy directly uses the absolute depth readings to constrain the pose estimations. This leads to a globally constrained scale in the estimation process as the pressure sensor measurements do not vary as a function of time (at least not within the short timelapses during which we use them).

However, it completely neglects the misalignment effect between the camera and the depth axis. Therefore, the second fusion strategy we propose takes into account this misalignment.

5.3.3.2 Strategy 2: Integration of relative depth measurements

In order to limit the misalignment effect, we propose to instead apply relative depth constraints between camera's poses instead of absolute ones. This will reduce the error due to the misalignment of the camera with respect to the depth axis because, given the low dynamic of the motions performed by the robot and the relatively high frequency of pressure sensors (> 5 Hz) and of

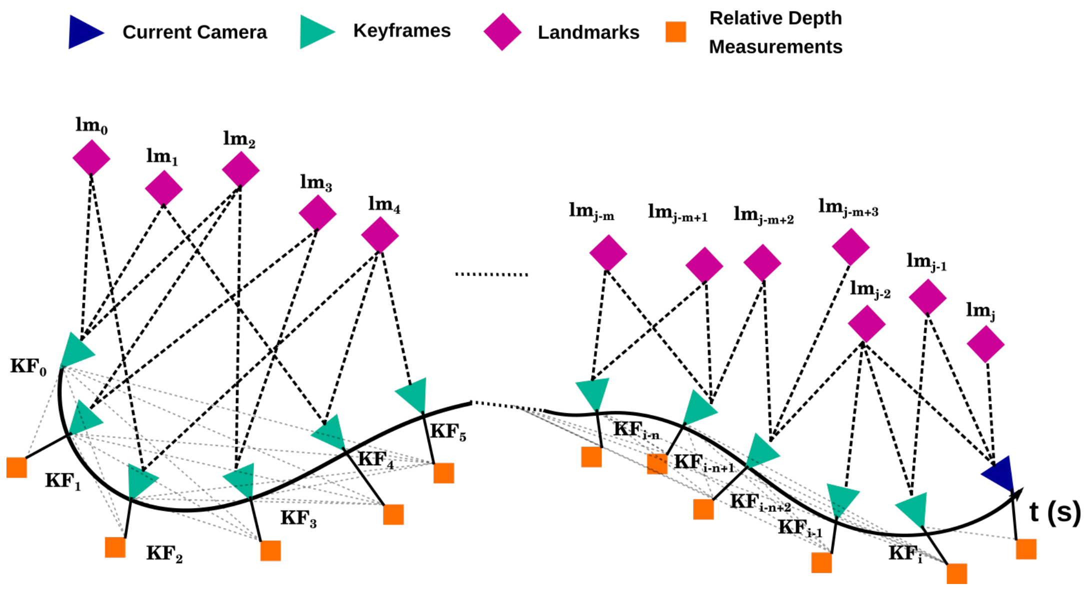


FIGURE 5.4: Strategy 2: Visual-Pressure SLAM with relative depth measurements. This schema represent the factor graph that can be derived from this SLAM problem. The states to optimize are the poses of the keyframes and of the current frame along with the visual landmarks. The different states are bound together by visual measurements (dark dotted lines) and relative depth measurements (orange squares with light dotted lines).

the camera (around 20 Hz in our case), the relative depth difference between two cameras' poses will stay low.

From Eq.(5.6), we model the effect of using a relative depth measurement between $\mathbf{X}_i$ and $\mathbf{X}_j$:

$$\bar{d}_{ij} = \bar{d}_j - \bar{d}_i \tag{5.14}$$

$$\bar{d}_{ij} = \bar{t^z}_{c_0c_j} \cdot \cos(\alpha) - \bar{t^z}_{c_0c_i} \cdot \cos(\alpha) \tag{5.15}$$

$$\bar{d}_{ij} = \left(\bar{t^z}_{c_0c_j} - \bar{t^z}_{c_0c_i}\right) \cdot \cos(\alpha) \tag{5.16}$$

where C_0 denotes the frame of reference. The resulting error due to misalignment will therefore be most of the time lower than when using absolute measurements (unless in the rare case where $\bar{t^z}_{c_0c_j} < |\bar{t^z}_{c_0c_j} - \bar{t^z}_{c_0c_i}|$).

As in the first strategy, the relative depth error terms are used both in pose estimation at frame-rate and in the Bundle Adjustment part.

Frame-rate Pose Estimation

For the pose estimation at frame rate, we compute a first estimate of the camera's pose by means of the P3P method. Then, we extend the pose refinement step by linking the current frame to the previous keyframe. The refinement of the pose can then be written as:

$$\mathbf{X}_i^* = \arg\min_{\mathbf{X}_i} \left(E_{visual}\left(\mathbf{X}_i\right) + \beta \cdot E_{rel\text{-}depth}\left(\mathbf{X}_k, \mathbf{X}_i\right) \right) \tag{5.17}$$

with β defined as in Eq.(5.9) and $E_{rel\text{-}depth}\left(\mathbf{X}_k, \mathbf{X}_i\right)$ defined as:

$$E_{rel\text{-}depth}\left(\mathbf{X}_k, \mathbf{X}_i\right) = \left(\| \left(\tilde{d}_i - \tilde{d}_k\right) - \left(\hat{t^z}_{Wc_i} - \hat{t^z}_{Wc_k}\right) \|^2_{\left(2 \cdot \sigma_{depth}\right)^2} \right) \tag{5.18}$$

The previous keyframe pose $\mathbf{X}_k$ and the set of landmarks $\mathfrak{L}_i$ observed from the current pose $\mathbf{X}_i$ are kept fixed during this optimization.

Depth-enhanced Bundle Adjustment

For the enhancement of the Bundle Adjustment, we could similarly add a relative depth error term between two consecutive keyframes. However, most of the time, the motion along the z-axis between consecutive keyframes is small and the resulting depth measurement is below the pressure sensor measurements' noise. Such measurements will be useful to constrain the localization estimations but, during the initialization phase, they will not bear any meaningful information to estimate the scale factor. Therefore, in order to maximize the chance of having significant depth variations, we create a relative depth measurement between each keyframe and its 10 preceding keyframes.

This new strategy is illustrated by Fig 5.4. The visual part of the graph stays identical to what it was in the vision-only problem but the graph is enhanced by adding depth factors between consecutive keyframes. Moreover, for the pose estimation at frame-rate, a depth factor is also added between the last keyframe and the current frame.

Following this new graph formulation, we define the new Bundle Adjustment problem to solve as:

$$\chi^* = \arg\min_{\chi} \left(E_{visual}\left(\chi\right) + E_{rel\text{-}depth}\left(\chi\right) \right) \tag{5.19}$$

where $E_{rel\text{-}depth}\left(\chi\right)$ is the energy term related to the relative depth factors that we write as:

$$\begin{aligned} E_{rel\text{-}depth}(\chi) &= \sum_{i \in \mathfrak{K}^*} \sum_{j=i-1}^{i-10} E_{rel\text{-}depth}\left(\mathbf{X}_j, \mathbf{X}_i\right) \\ &= \sum_{i \in \mathfrak{K}^*} \sum_{j=i-1}^{i-10} \left(\| \left(\tilde{d}_i - \tilde{d}_j\right) - \left(\hat{t^z}_{WC_i} - \hat{t^z}_{WC_j}\right) \|^2_{\left(2 \cdot \sigma_{depth}\right)^2} \right) \end{aligned} \tag{5.20}$$

where $\mathfrak{K}^*$ is the set of keyframes in χ but without including the first one, $\hat{t^z}_{WC_i}$ is the SLAM estimated position of the i'th keyframe with respect to the z axis of the reference frame and $\tilde{d}_i$ is the depth measured at keyframe i. The same notation holds for the keyframe j. Note also that the variance of the depth error term is set to $\left(2 \cdot \sigma_{depth}\right)^2$ to account for the two measurements used to defined the error term.

We note that linking each keyframe to the ten previous ones will also slightly increase the misalignment error compared to linking only two consecutive keyframes. However, it should also better prevent the localization estimations from drifting compared to simply constraining the position along the z-axis with respect to the previous keyframe. This trade-off between accuracy and consistency has been chosen empirically.

5.3.4 Visual-Pressure Monocular Initialization

To improve the convergence of the estimations toward the true scale of the trajectory, we also apply several modifications to the initialization procedure of UW-VO. First, we ensure performing the visual initialization between two frames linked to a depth measurement. This will serve in the scale factor initialization detailed next.

5.3.4.1 Scale Factor Initialization

After extracting the relative rotation and translation between these two frames, as described in 4.5.7 (page 79), we scale the translation with the depth measurement related to the second frame $\mathbf{X}_1$:

$$s = \frac{\tilde{d}_1}{\hat{t^z}_{WC_1}} \quad \text{and} \quad {}^{scaled}\hat{t}_{WC_1} = s \cdot \hat{t}_{WC_1} \tag{5.21}$$

However, due to the sensor noise, potential low motion along the depth axis and misalignment error, this first estimated scaled factor s might still be far from the truth. In order to ensure a fast convergence towards the true scale, we additionally modify the Bundle Adjustment steps.

In UW-VO, the cost function defined for the Bundle Adjustment is invariant to global change in translation, rotation and of the scale (see section 4.6.3). The resulting 7DOF possible drift from the frame of reference has to be handled by fixing at least two keyframes when solving the Bundle Adjustment problem. This process is known as gauge fixing [216]. However, scale is now observable through the depth measurements and the gauge can be fixed by simply keeping one keyframe fixed during optimization. Therefore, we do not fix any keyframe's pose but the first one in the Bundle Adjustment until a certain quantity N of keyframes has been reached. This allows a smooth convergence of the scale and enough motions to relate the visual measurements to the depth ones. Indeed, if there is almost no vertical motion, the scale will not be recoverable. Hence, if we were to follow the exact same strategy as in the pure Visual SLAM methods, that is fixing at least two keyframes in the problem and more if older keyframes are observing currently optimized 3D landmarks, the global scale of the trajectory would be highly constrained by the fixed states.

In practice, using $N = 30$, we obtain good scale estimations. Once, the N first keyframes created, we turn back to the local adaptive windowed strategy employed in the Visual SLAM (see sec. 4.6.3).

5.3.4.2 Camera - Depth Alignement by Relaxation of the Gauge Constraint

In order to tackle the camera-depth alignment issue, we have also investigated another strategy that further improves the initialization procedure detailed above.

It appears that the depth measurements render the depth axis observable and can in fact be used as a reference frame that would be aligned with this axis. Therefore, we propose to relax the gauge constraint over the orientation of the frame of reference in the Bundle Adjustment steps taken during the initialization phase. As the visual pose estimations are always relative to a frame of reference (the first keyframe here), relaxing the gauge constraint over the orientation of this first keyframe will allow it to *move* in order to maximize the likelihood of the depth measurements given the current visual estimations.

In other words, initially the first keyframe defined the frame of reference and was therefore represented by a zero vector for its position and an identity rotation matrix. In order to keep this frame of reference (or gauge) fixed

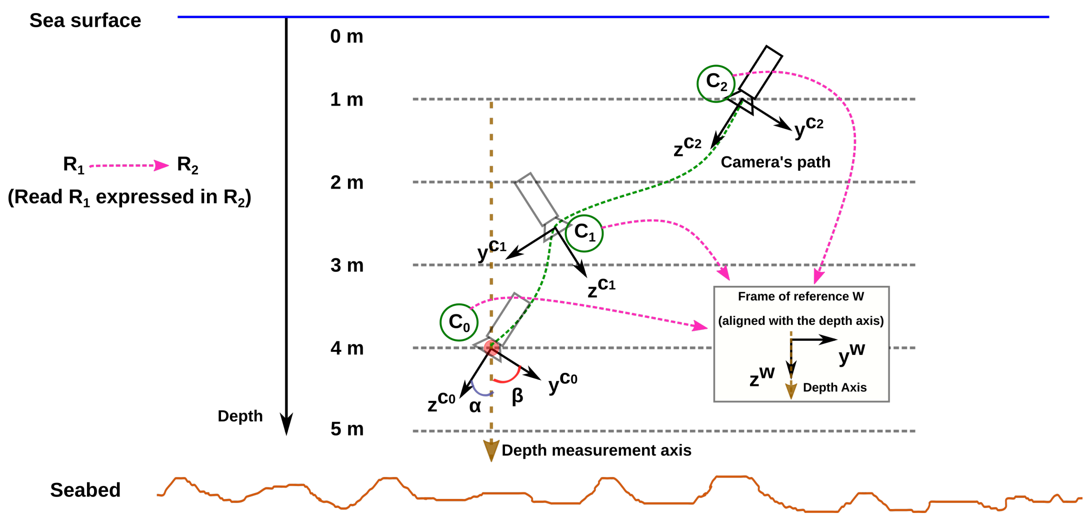


FIGURE 5.5: Difference between depth measurements and the frame of reference of camera's poses when relaxing the gauge constraint. C_0, C_1 and C_2 represent the camera's frame at time 0, 1 and 2. C_0 is the initial frame but not the frame of reference in this case. The orientation of the frame of reference is freed by the gauge relaxation which allows it to align with the depth axis during the initialization phase.

during the Bundle Adjustment steps, the first keyframe included in the optimization was kept fixed. What we now propose is to only fix the position of the first keyframe and not its orientation, which is included in the states to estimate.

In practice, this relaxation leads to a quick convergence towards a solution that scales the estimated trajectory to the metric scale. What happens here is that the scaling is done not only by fitting the norm of the visual position estimations with the depth measurements, but also by rotating the frame of reference in a way that aligns its z axis to the depth axis (see Fig. 5.5).

5.3.5 Experimental Results

We now present the results obtained with the different strategies proposed. Moreover, we compare these strategies using three different configurations for the algorithm:

- Regular: In this mode, the initialization is performed as explained in section 5.3.4.1 and the optimization steps are then performed normally.
- Free Gauge: In this mode, we relax the gauge constraint during the initialization phase as described in section 5.3.4.2.

- IMU Aligned: In this mode, we use the accelerometer measurements from an IMU to align the first keyframe z-axis, that is the frame of reference, with gravity.

The IMU alignment step is performed by averaging the first 50 accelerometer measurements to estimate the gravity axis and we then compute the rotation matrix that rotates the frame of reference such that its z-axis gets aligned with the gravity axis. This alignment procedure is not perfect, but is mainly used to analyze the misalignment effects by comparing it to the regular mode. Moreover, the low dynamics of ROVs ensure that the ROV's own acceleration will not influence too much the accelerometer measurements, resulting in relatively low errors when estimating the gravity axis.

We also include results obtained by using UW-VO, the vision-only SLAM system presented in chapter 4, and simply scaling its first estimated translation using the depth measurements (we refer to it as *Init. Only* in what follows).

We evaluate the different strategies and configurations on sequences extracted from the AQUALOC dataset, which is actually composed of two datasets recorded with a different hardware.

The first dataset has been recorded in a harbor, in shallow waters, and its measurements come from a fisheye-modeled camera and from a pressure sensor with 2 mm of resolution and an acquisition frequency of 5 Hz. The camera is installed approximately vertically on the ROV and is oriented towards the seabed. A light work class ROV was used here, leading to quite dynamic sequences.

The second dataset has been recorded on archaeological sites, in deep waters ($\approx$ 400 meters), and its measurements come from a pinhole-modeled camera and from a pressure sensor with 3 cm of resolution and an acquisition frequency of 60 Hz. In this dataset, the camera is placed with an approximate angle of 30 degrees with respect to the vertical axis. More detailed information about these datasets are given in chapter 6. A work class ROV was used here, well stabilized and with very slow motions.

Both these datasets contain a comparative baseline trajectory computed offline by means of photogrammetry and scaled with the depth measurements (more details in the next chapter). As in the previous chapter, we use the Absolute Trajectory Error (ATE) [212] as the metric to evaluate the accuracy of the estimated trajectories.

TABLE 5.1: Absolute trajectory errors (RMSE in m) on the Harbor dataset.

		Absolute Trajectory Error (m)						
		Init. Only	*Regular*		*Free Gauge*		*IMU Aligned*	
Seq.	Length (m)	UW-VO	Strat. 1	Strat. 2	Strat. 1	Strat. 2	Strat. 1	Strat. 2
# 1	39.3	1.01	0.55	0.53	0.56	0.49	0.52	0.64
# 2	75.6	1.70	1.23	0.40	1.08	0.36	0.63	0.36
# 3	23.6	0.52	0.30	0.26	0.23	0.25	0.25	0.24
# 5	28.5	0.96	0.19	0.11	0.12	0.13	0.15	0.13
# 6	19.5	0.17	0.11	0.06	0.04	0.04	0.04	0.05

Results on the first dataset

Table 5.1 displays the absolute trajectory errors obtained on the first dataset. On this dataset, as the camera is already oriented vertically, that is with a small angle between its z-axis and the depth axis, the different configurations do not influence much the accuracy of most of the estimated trajectories. Yet, when using absolute depth measurements (strategy 1), there is a noticable improvement when the misalignment issue is taken into account. We can also see that the results obtained with the second strategy are the best in the regular configuration and are equivalent in the free gauge and IMU align modes. This highlights the fact that using relative depth measurements is more robust to any potential misalignment between the camera and the depth axis, even when the misalignment is small.

Furthermore, we can see that fusing the pressure sensor measurements within the SLAM algorithm very efficiently recovers the scale of the trajectories and improves the localization accuracy compared to UW-VO. Note that no results are displayed for the sequences 4 and 7, because in both these sequences the visual information become unusable at some point. The pressure sensor alone is not sufficient to estimate the motions performed by the camera and hence the algorithm is not able to recover from such disturbances.

Results on the second dataset

Table 5.2 displays the absolute trajectory errors obtained on the second dataset. On this dataset, because there is a big misalignment between the camera's z-axis and the depth axis (about 30 degrees), the influence of the

TABLE 5.2: Absolute trajectory errors (RMSE in m) on the Archaeological dataset.

		Absolute Trajectory Error (m)						
		Init. Only	*Regular*		*Free Gauge*		*IMU Aligned*	
Seq.	Length (m)	UW-VO	Strat. 1	Strat. 2	Strat. 1	Strat. 2	Strat. 1	Strat. 2
# 8	41.2	0.96	0.58	0.52	0.44	0.34	0.41	0.53
# 9	65.4	1.3	0.99	0.90	1.01	0.72	0.72	0.71

different strategies and configurations is well highlighted. First, in the regular setup, the second strategy performs better than the first one. This reflects that using relative depth measurements is more efficient than using absolute ones when there exists a strong misalignment between the camera and the depth axis. Second, in the free gauge configuration, the second strategy also performs better. It seems that the optimization behaves more consistently in the initialization phase with the second strategy but this should be further investigated in future works. Third, in the IMU aligned mode, the first strategy obtains more accurate results in average. As the gravity alignment method employed is simplistic here, some misalignment might still exists in this case. However, the first strategy directly benefits from the alignment correction while the second strategy estimations are closer to the ones obtained in the regular setup, highlighting the fact that taking relative depth measurements reduces the misalignment effects. In all the different cases, we can also see that fusing the depth measurements improves the accuracy of the estimated trajectories compared to UW-VO.

5.3.6 Discussion

These results show that a simple pressure sensor can provide all the information required for recovering the scale of the trajectories. However, cares has to be taken when fusing depth measurements within a Visual SLAM algorithm because of potential misalignment between the camera and the depth axis. Among the proposed strategies, the best results are obtained when using relative depth measurements and relaxing the gauge constraint over the orientation of the frame of reference during the initialization phase, suggesting that this would be the optimal configuration for tightly fusing vision and pressure in a SLAM framework.

While the depth measurement provides a very useful information for the estimation of accurate trajectories, the localization is still completely dependent on the visual information and any strong visual disturbance leads to failure. In order to be robust to short loss of visual information, we next investigate the fusion of a MEMS-IMU within the Visual-Pressure localization framework.

5.4 Visual-Inertial-Pressure SLAM

We now investigate the tight fusion of an MEMS-IMU sensor within the Visual-Pressure SLAM algorithm described in the previous section. By doing so, we expect to further improve the localization accuracy and to gain in robustness, especially during short loss of visual information.

5.4.1 Inertial Measurement Unit Sensor

A typical MEMS-IMU is composed of a three axes gyroscope and a three axes accelerometer. The gyroscope measures the angular velocity of the sensor and the accelerometer measures the force compensating its free-fall.

In this section, we investigate the fusion of a low-cost MEMS-IMU within the SLAM framework. Such IMUs are not accurate enough to measure the Coriolis acceleration and the magnitude of the earth rotation. The gyroscope and accelerometer measurements can hence be modeled as follows [130]:

$$\tilde{\boldsymbol{\omega}}_B(t) = \boldsymbol{\omega}_B(t) + \mathbf{b}^g(t) + \boldsymbol{\eta}^g \tag{5.22}$$

$$\tilde{\mathbf{a}}_B(t) = \mathbf{R}_{WB}(t)^T \cdot (\mathbf{a}_W(t) - \mathbf{g}_W) + \mathbf{b}^a(t) + \boldsymbol{\eta}^a \tag{5.23}$$

where $\tilde{\boldsymbol{\omega}}_B \in \mathbb{R}^3$ and $\tilde{\boldsymbol{a}}_B \in \mathbb{R}^3$ represent the outputs of the IMU sensor, $\mathbf{R}_{WB} \in \mathbb{SO}(3)$ is the orientation of the IMU sensor with respect to the World frame, $\boldsymbol{\omega}_B \in \mathbb{R}^3$ and $\boldsymbol{a}_B \in \mathbb{R}^3$ represent the actual angular velocity and linear acceleration applied to the IMU in the world frame, $\mathbf{b}^g \in \mathbb{R}^3$ and $\mathbf{b}^a \in \mathbb{R}^3$ are the gyroscope and accelerometer biases and $\boldsymbol{\eta}^g \in \mathbb{R}^3$ and $\boldsymbol{\eta}^a \in \mathbb{R}^3$ models the IMU noise.

Here, we consider a simplified IMU model where we neglect the potential issues about the misalignment and scale errors of the accelerometer and gyroscope axes. The noise terms are modeled as zero-mean gaussian distributions:

$$\boldsymbol{\eta}^g \sim \mathcal{N}(\mathbf{0}_{3\times1}, \mathbf{I}_{3\times3} \cdot \sigma_g^2) \tag{5.24}$$

$$\boldsymbol{\eta}^a \sim \mathcal{N}(\mathbf{0}_{3\times1}, \mathbf{I}_{3\times3} \cdot \sigma_a^2) \tag{5.25}$$

where σ_g and σ_a are estimated by calibrating the IMU and computing the Allan Deviation [58].

5.4.2 Problem Statement

The task at hand here is to integrate the measurements from a low-cost IMU and a pressure sensor to the Visual SLAM algorithm. In order to keep the NLLS form of our localization solution and to make the vision part benefit from these other sensing modalities, we propose to tightly fuse the IMU and pressure measurements to the visual data. As the SLAM problem can be formulated with a factor graph, fusing new sensors measurements within the SLAM problem is equivalent to adding new factors and nodes within the graph.

The IMU's gyroscope measures the angular velocity, which can be integrated to provide rotational constraints between two consecutive pose nodes. The IMU's accelerometer gives an information about the linear acceleration applied to it. Constraining the position differences between two consecutive nodes thus requires double integrating the accelerometer measurements, first estimating the velocity and then estimating the position. As the velocity is going to be an intermediate state of interest, a velocity node has to be added to the graph. Furthermore, the gyroscopes and accelerometers measurements are influenced by time-varying biases which have to be estimated as well.

More formally, the navigational state's parameters we want to estimate at each time step in this Visual-Inertial-Pressure SLAM system are:

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{R}_{WBi} & \mathbf{p}_{WBi} & \mathbf{v}_{WBi} & \mathbf{b}_i^g & \mathbf{b}_i^a \end{bmatrix}^T \tag{5.26}$$

where $\mathbf{R}_{WBi} \in \mathbb{SO}(3)$ is the body orientation with respect to the World frame, $\mathbf{p}_{WBi} \in \mathbb{R}^3$ is the body position, $\mathbf{v}_{WBi} \in \mathbb{R}^3$ is the body velocity and $\mathbf{b}_i^g \in \mathbb{R}^3$ and $\mathbf{b}_i^a \in \mathbb{R}^3$ are time-varying biases of the gyroscope and accelerometer, respectively. The body frame B is centered on the IMU as classically done for Visual-Inertial navigation systems and is used as the new coordinate frame (instead of using the camera's coordinate frame as in the VSLAM and Visual-Pressure SLAM methods).

In order to estimate these states within a NLLS solver, we define the following optimization problem:

$$\chi^* = \underset{\chi}{\arg\min} \left(E_{visual}(\chi) + E_{imu}(\chi) + E_{depth}(\chi) \right) \tag{5.27}$$

The update to apply to the navigational state $\delta\mathbf{X}_i \in \mathbb{R}^{15}$ during the optimization steps are defined as:

$$\mathbf{X}_i^* = \mathbf{X}_i \boxplus \delta\mathbf{X}_i \tag{5.28}$$

In the context of Visual-Inertial SLAM, NLLS optimizations have to be performed more frequently than in Visual SLAM in order to keep low errors when integrating the IMU measurements (especially by estimating the IMU biases required to correct the IMU measurements). Hence, instead of defining the pose states $(\mathbf{R}_{WBi}, \mathbf{p}_{WBi}) \in \mathbb{SE}(3)$ and having to compute the potentially expensive Jacobian $\mathbf{V}(\delta\mathbf{R})$ (see section 2.7 page 34), we define the pose as a *pseudo*-$\mathbb{SE}(3)$ [18]. The pose state updates are therefore expressed as:

$$\begin{aligned} \mathbf{R}_{WBi}^* &= \mathbf{R}_{WBi} \boxplus \delta\mathbf{R}_i = \mathbf{R}_{WBi} \cdot \text{Exp}(\delta\mathbf{R}_i) \\ \mathbf{p}_{WBi}^* &= \mathbf{p}_{WBi} \boxplus \delta\mathbf{p}_i = \mathbf{p}_{WBi} + \mathbf{R}_{WBi} \cdot \delta\mathbf{p}_i \end{aligned} \tag{5.29}$$

where $\text{Exp}(\cdot)$ is the exponential mapping from $\mathbb{R}^3$ to the Special Orthogonal group $\mathbb{SO}(3)$ defined as in section 2.7 (page 34).

The remaining state parameters belonging to $\mathbb{R}^3$, their updates are defined by classical addition:

$$\begin{aligned} \mathbf{v}_{WBi}^* &= \mathbf{v}_{WBi} \boxplus \delta\mathbf{v}_i = \mathbf{v}_{WBi} + \delta\mathbf{v}_i \\ \mathbf{b}_i^{g*} &= \mathbf{b}_i^g \boxplus \delta\mathbf{b}_i^g = \mathbf{b}_i^g + \delta\mathbf{b}_i^g \\ \mathbf{b}_i^{a*} &= \mathbf{b}_i^a \boxplus \delta\mathbf{b}_i^a = \mathbf{b}_i^a + \delta\mathbf{b}_i^a \end{aligned} \tag{5.30}$$

The next objective for solving the Visual-Inertial-Pressure SLAM problem is to define the different error terms of the cost function (Eq.(5.27)).

5.4.3 State Parameters Evolution

In order to link the IMU measurements to the state parameters, we define the evolution of the state $\mathbf{X}$ as following this kinematic model:

$$\dot{\mathbf{R}}_{WB} = \mathbf{R}_{WB} \cdot \boldsymbol{\omega}_B^{\wedge} \;, \qquad \dot{\mathbf{p}}_{WB} = \mathbf{v}_{WB} \;, \qquad \dot{\mathbf{v}}_{WB} = \mathbf{a}_{WB} \tag{5.31}$$

where $(\cdot)^{\wedge}$ is the skew-symmetric matrix operator.

The state at time $t + \Delta t$ can be obtained by integrating Eq.(5.31):

$$\begin{aligned}
\mathbf{R}_{WB}(t+\Delta t) &= \mathbf{R}_{WB}(t) \cdot \exp\left(\int_t^{t+\Delta t} \boldsymbol{\omega}_B(\tau)^{\wedge} d\tau\right) \\
\mathbf{v}_{WB}(t+\Delta t) &= \mathbf{v}_{WB}(t) + \int_t^{t+\Delta t} \mathbf{a}_W(t) d\tau \\
\mathbf{p}_{WB}(t+\Delta t) &= \mathbf{p}_{WB}(t) + \int_t^{t+\Delta t} \mathbf{v}_{WB}(\tau) d\tau + \int\int_t^{t+\Delta t} \mathbf{a}_W(\tau) d\tau^2
\end{aligned} \tag{5.32}$$

Assuming that $\mathbf{a}_W(\tau)$ and $\boldsymbol{\omega}_B(\tau)$ are constant in $\tau \in [t, t+\Delta t]$, we can solve Eq.(5.32) in discrete-time using Euler integration method:

$$\begin{aligned}
\mathbf{R}_{WB}(t+\Delta t) &= \mathbf{R}_{WB}(t) \cdot \mathrm{Exp}\left(\boldsymbol{\omega}_B(t) \cdot \Delta t\right) \\
\mathbf{v}_{WB}(t+\Delta t) &= \mathbf{v}_{WB}(t) + \mathbf{a}_W(t) \cdot \Delta t \\
\mathbf{p}_{WB}(t+\Delta t) &= \mathbf{p}_{WB}(t) + \mathbf{v}_{WB}(t) \cdot \Delta t + \frac{1}{2} \cdot \mathbf{a}_W(t) \cdot \Delta t^2
\end{aligned} \tag{5.33}$$

Linking Eq.(5.33) to the IMU measurements model Eq.(5.22-5.23), we obtain:

$$\begin{aligned}
\mathbf{R}_{WB}(t+\Delta t) &= \mathbf{R}_{WB}(t) \cdot \mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_B(t) - \mathbf{b}^g(t) - \boldsymbol{\eta}^g) \cdot \Delta t\right) \\
\mathbf{v}_{WB}(t+\Delta t) &= \mathbf{v}_{WB}(t) + \mathbf{g}_W \cdot \Delta t + \mathbf{R}_{WB}(t) \cdot (\tilde{\mathbf{a}}_B(t) - \mathbf{b}^a(t) - \boldsymbol{\eta}^a) \cdot \Delta t \\
\mathbf{p}_{WB}(t+\Delta t) &= \mathbf{p}_{WB}(t) + \mathbf{v}_{WB}(t) \cdot \Delta t + \frac{1}{2} \cdot \mathbf{g}_W \cdot \Delta t^2 \\
&\qquad + \frac{1}{2} \cdot \mathbf{R}_{WB}(t) \cdot (\tilde{\mathbf{a}}_B(t) - \mathbf{b}^a(t) - \boldsymbol{\eta}^a) \cdot \Delta t^2
\end{aligned} \tag{5.34}$$

The biases are modeled as random-walk processes:

$$\dot{\mathbf{b}_{\mathbf{g}}} = \mathbf{0}_{3\times 1} \;, \qquad \dot{\mathbf{b}_{\mathbf{a}}} = \mathbf{0}_{3\times 1} \tag{5.35}$$

$$\mathbf{b}^g(t+\Delta t) = \mathbf{b}^g(t) + \boldsymbol{\eta}_{b_g} \tag{5.36}$$
$$\mathbf{b}^a(t+\Delta t) = \mathbf{b}^a(t) + \boldsymbol{\eta}_{b_a} \tag{5.37}$$
$$\boldsymbol{\eta}_{b_g} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \mathbf{I}_{3\times3} \cdot \sigma^2_{b_g}) \quad , \quad \boldsymbol{\eta}_{b_a} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \mathbf{I}_{3\times3} \cdot \sigma^2_{b_a}) \tag{5.38}$$

In discrete time, we can model our system as following this state propagation:

$$\begin{aligned}
\mathbf{R}_{WB}(k+1) &= \mathbf{R}_{WB}(k) \cdot \mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_B(k) - \mathbf{b}^g(k)) \cdot \Delta t_{kk+1}\right) \\
\mathbf{v}_{WB}(k+1) &= \mathbf{v}_{WB}(k) + \mathbf{g}_W \cdot \Delta t_{kk+1} + \mathbf{R}_{WB}(k) \cdot (\tilde{\mathbf{a}}_B(k) - \mathbf{b}^a(k)) \cdot \Delta t_{kk+1} \\
\mathbf{p}_{WB}(k+1) &= \mathbf{p}_{WB}(k) + \mathbf{v}_{WB}(k) \cdot \Delta t_{kk+1} + \frac{1}{2} \cdot \mathbf{g}_W \cdot \Delta t^2_{kk+1} \\
&\qquad + \frac{1}{2} \cdot \mathbf{R}_{WB}(k) \cdot (\tilde{\mathbf{a}}_B(k) - \mathbf{b}^a(k)) \cdot \Delta t^2_{kk+1} \\
\mathbf{b}^g(k+1) &= \mathbf{b}^g(k) \\
\mathbf{b}^a(k+1) &= \mathbf{b}^a(k)
\end{aligned} \tag{5.39}$$

where $\Delta t_{kk+1} = t_{k+1} - t_k$, *i.e.* Δt_{kk+1} is the time interval between two IMU measurements.

Note that the derivation of this model is based on the approximation that $\mathbf{a}_W(\tau)$ and $\boldsymbol{\omega}_B(\tau)$ are constant within a time interval. The accuracy of this approximation is hence dependant of the length of this time interval. In the context of this thesis, we have mainly worked with IMU measurements produced at 200 Hz and more. Such high rates ensure that the constant approximation over the angular velocity and linear acceleration measurements is realistic enough. However, when using low-rates IMU, high order methods such as Runge-Kutta integrations should be used [34].

5.4.4 IMU Preintegration

The discrete state evolution equations (5.34) define how the IMU measurements interact with the states of interest. However, adding IMU measurements into the factor graph at the IMU rate rapidly becomes intractable as they would have to be re-computed for each update of their related bias value during optimization [148].

The way to deal with these high rates measurements is to turn them into *preintegrated* IMU measurements [148]. The idea of preintegration is to accumulate the IMU measurements between two visual measurements (*i.e.* between two images) in order to provide single measurements linking the states related to the visual measurement times. In this case, the nodes of the factor graph are the states at the camera-rate (*i.e.* exactly as in the Visual SLAM formulation). Furthermore, in addition to the factors representing the 3D landmarks observations, each node is linked to the previous one and to the next one by a preintegrated IMU factor.

The IMU preintegration technique has been first proposed by [148]. This technique was applied using Euler angles and was then extended to the quaternion form in [183, 6] and to the $\mathbb{SO}(3)$ manifold in [80]. As the Visual SLAM method was based on a manifold formulation by using states defined on $\mathbb{SE}(3)$ and directly optimizing on this manifold space, we follow the preintegration theory developed in [80].

Taking Eq.(5.34) and assuming the camera and IMU measurements perfectly synchronized and constant IMU biases, we can model the relative motion between image i and image j as:

$$\begin{aligned}
\mathbf{R}_{WBj} &= \mathbf{R}_{WBi} \cdot \prod_{k=i}^{j-1} \cdot \operatorname{Exp}\left((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k^g - \boldsymbol{\eta}^g) \cdot \Delta t_{kk+1} \right) \\
\mathbf{v}_{WBj} &= \mathbf{v}_{WBi} + \mathbf{g}_W \cdot \Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{R}_{WB_k} \cdot (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}^a) \cdot \Delta t_{kk+1} \\
\mathbf{p}_{WBj} &= \mathbf{p}_{WBi} + \frac{1}{2} \cdot \mathbf{g}_W \cdot \Delta t_{ij}^2 \\
&\quad + \sum_{k=i}^{j-1} \left[\mathbf{v}_{WB_k} \cdot \Delta t_{kk+1} + \frac{1}{2} \cdot \mathbf{R}_{WB_k} \cdot (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}^a) \cdot \Delta t_{kk+1}^2 \right]
\end{aligned} \tag{5.40}$$

with $\Delta t_{ij} = t_j - t_i$, where t_i and t_j are respectively the timestamps of the images i and j.

The sum and product terms of Eq.(5.40) contain all the IMU measurements and could hence be used as preintegrated measurements that could be readily added to our factor graph. However, if we directly use this formulation, the preintegration would have to be recomputed every time the estimates of the rotational component of the states $\mathbf{R}_{WBk}$ is updated. This is due to the fact that, here, the preintegrated measurements are defined as relative motions in the World frame. In order to avoid repeating the computation of the preintegrated measurements, we instead express them as relative motions in the body frame B_i:

$$\begin{aligned}
\Delta\tilde{\mathbf{R}}_{BiBj} &\doteq \mathbf{R}_{BiW} \cdot \mathbf{R}_{WBj} \\
&= \prod_{k=i}^{j-1} \cdot \operatorname{Exp}\left(\left(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k^g - \boldsymbol{\eta}^g\right) \cdot \Delta t_{kk+1}\right) \\
\Delta\tilde{\mathbf{v}}_{BiBj} &\doteq \mathbf{R}_{BiW} \cdot \left(\mathbf{v}_{WBj} - \mathbf{v}_{WBi} - \mathbf{g}_W \cdot \Delta t_{ij}\right) \\
&= \sum_{k=i}^{j-1} \Delta\mathbf{R}_{BiB_k} \cdot (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}^a) \cdot \Delta t_{kk+1} \\
\Delta\tilde{\mathbf{p}}_{BiBj} &\doteq \mathbf{R}_{BiW} \cdot \left(\mathbf{p}_{WBj} - \mathbf{p}_{WBi} - \mathbf{v}_{WBi} \cdot \Delta t_{ij} - \frac{1}{2} \cdot \mathbf{g}_W \cdot \Delta t_{ij}^2\right) \\
&= \sum_{k=i}^{j-1} \left[\Delta\mathbf{v}_{WB_k} \cdot \Delta t_{kk+1} + \frac{1}{2} \cdot \Delta\mathbf{R}_{BiB_k} \cdot (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a - \boldsymbol{\eta}^a) \cdot \Delta t_{kk+1}^2\right]
\end{aligned} \tag{5.41}$$

With this new formulation, the preintegrated measurements do not depend on the rotational states anymore. Putting apart the variations due to varying biases for now, these preintegrated measurements are hence constant over time and can be considered as normally distributed random variables, whose expectations are:

$$\begin{aligned}
\Delta\mathbf{p}_{BiB_{k+1}} &= \Delta\mathbf{p}_{BiBk} + \Delta\mathbf{v}_{BiBk} \cdot \Delta t_{kk+1} + \Delta\mathbf{R}_{BiB_k} \cdot \delta\mathbf{p}_{B_kB_{k+1}} \\
\Delta\mathbf{v}_{BiB_{k+1}} &= \Delta\mathbf{v}_{BiBk} + \Delta\mathbf{R}_{BiB_k} \cdot \delta\mathbf{v}_{B_kB_{k+1}} \\
\Delta\mathbf{R}_{BiB_{k+1}} &= \Delta\mathbf{R}_{BiBk} \cdot \delta\mathbf{R}_{B_kB_{k+1}}
\end{aligned} \tag{5.42}$$

This new formulation defines how the preintegration measurements are updated upon reception of a new IMU measurement, the preintegration increments being defined as:

$$\begin{aligned}
\delta\mathbf{p}_{B_kB_{k+1}} &= \frac{1}{2} \cdot (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a) \cdot \Delta t_{kk+1}^2 \\
\delta\mathbf{v}_{B_kB_{k+1}} &= (\tilde{\mathbf{a}}_k - \mathbf{b}_k^a) \cdot \Delta t_{kk+1} \\
\delta\mathbf{R}_{B_kB_{k+1}} &= \operatorname{Exp}\left(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k^g\right)
\end{aligned} \tag{5.43}$$

The computation of the preintegrated IMU measurements can be seen as the state prediction step of a Kalman Filter. In order to use these measurements efficiently within a NLLS solver, we have to estimate their covariance.

With a slight abuse of notation, we define the following state and error state:

$$\Delta\hat{\mathbf{I}}_{ik} = \begin{bmatrix} \Delta\mathbf{p}_{BiB_k} & \Delta\mathbf{v}_{BiB_k} & \Delta\mathbf{R}_{BiB_k} \end{bmatrix}^T \tag{5.44}$$

$$\delta\hat{\mathbf{I}}_{kk+1} = \begin{bmatrix} \delta\mathbf{p}_{B_kB_{k+1}} & \delta\mathbf{v}_{B_kB_{k+1}} & \delta\mathbf{R}_{B_kB_{k+1}} \end{bmatrix}^T \tag{5.45}$$

The state prediction of the preintegrated measurements can then be written as:

$$\Delta\hat{\mathbf{I}}_{ik+1} = f\left(\Delta\hat{\mathbf{I}}_{ik}, \delta\hat{\mathbf{I}}_{kk+1}\right) \tag{5.46}$$

and the covariance $\mathbf{\Sigma}^{\Delta\mathbf{I}}_{ik+1}$ can be incrementally estimated using the jacobians of $f(\cdot)$:

$$\mathbf{\Sigma}^{\Delta\mathbf{I}}_{ik+1} = \frac{\partial f}{\partial\Delta\hat{\mathbf{I}}_{ik+1}} \cdot \mathbf{\Sigma}^{\Delta\mathbf{I}}_{ik} \cdot \frac{\partial f}{\partial\Delta\hat{\mathbf{I}}_{ik+1}}^T + \frac{\partial f}{\partial\boldsymbol{\eta}^{a,g}} \cdot \mathbf{\Sigma}^{\eta} \cdot \frac{\partial f}{\partial\boldsymbol{\eta}^{a,g}}^T \tag{5.47}$$

where $\mathbf{\Sigma}^{\eta}$ is the covariance associated to the gyroscope and accelerometer noises: $\boldsymbol{\eta}^{a,g} = \begin{bmatrix} \boldsymbol{\eta}^a & \boldsymbol{\eta}^g \end{bmatrix}^T$. The jacobian related to the IMU noise $\boldsymbol{\eta}^{a,g}$ can be computed from the chain rule:

$$\frac{\partial f}{\partial\boldsymbol{\eta}^{a,g}} = \frac{\partial f}{\partial\delta\hat{\mathbf{I}}_{kk+1}} \cdot \frac{\partial\delta\hat{\mathbf{I}}_{kk+1}}{\partial\boldsymbol{\eta}^{a,g}} \tag{5.48}$$

where the different jacobians are defined as:

$$\begin{aligned}
\frac{\partial f}{\partial\Delta\hat{\mathbf{I}}_{ik+1}} &= \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3}\cdot\Delta t_{kk+1} & \Delta\mathbf{R}_{BiB_k}\cdot\left(\delta\mathbf{p}_{B_kB_{k+1}}\right)^{\wedge} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \Delta\mathbf{R}_{BiB_k}\cdot\left(\delta\mathbf{v}_{B_kB_{k+1}}\right)^{\wedge} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{R}^T_{B_kB_{k+1}} \end{bmatrix} \in \mathbb{R}^{9\times9} \\
\frac{\partial f}{\partial\delta\hat{\mathbf{I}}_{kk+1}} &= \begin{bmatrix} \Delta\mathbf{R}_{BiB_k} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \Delta\mathbf{R}_{BiB_k} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{9\times9} \\
\frac{\partial\delta\hat{\mathbf{I}}_{kk+1}}{\partial\boldsymbol{\eta}^{a,g}} &= \begin{bmatrix} \frac{1}{2}\cdot\mathbf{I}_{3\times3}\cdot\Delta t^2_{kk+1} & \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3}\cdot\Delta t_{kk+1} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{V}\left(\left(\tilde{\omega}_k - \mathbf{b}^g_k\right)\cdot\Delta t_{kk+1}\right)\cdot\Delta t_{kk+1} \end{bmatrix} \in \mathbb{R}^{9\times6}
\end{aligned} \tag{5.49}$$

with $\mathbf{V}(\cdot)$ the left Jacobian of $\mathbb{SO}(3)$ (defined in section 2.7).

Last, we recall that until here, the IMU biases have been considered as being known and constant. This is not true and, as these biases are parts of the states we want to estimate, we have to define how slight changes in the biases values influence the preintegrated measurements. Since we want to avoid recomputing the preintegrated measurements when the states estimates are updated, the effects of modifying the biases are modeled through a first-order Taylor propagation:

$$\begin{aligned}
\Delta\tilde{\mathbf{R}}_{BiBj} &= \Delta\tilde{\mathbf{R}}_{BiBj} \cdot \mathrm{Exp}\left(\frac{\partial\Delta\tilde{\mathbf{R}}_{BiBj}}{\partial\mathbf{b}^g} \cdot \delta\mathbf{b}^g\right) \\
\Delta\tilde{\mathbf{v}}_{BiBj} &= \Delta\tilde{\mathbf{v}}_{BiBj} + \frac{\partial\Delta\tilde{\mathbf{v}}_{BiBj}}{\partial\mathbf{b}^g} \cdot \delta\mathbf{b}^g + \frac{\partial\Delta\tilde{\mathbf{v}}_{BiBj}}{\partial\mathbf{b}^a} \cdot \delta\mathbf{b}^a \\
\Delta\tilde{\mathbf{p}}_{BiBj} &= \Delta\tilde{\mathbf{p}}_{BiBj} + \frac{\partial\Delta\tilde{\mathbf{p}}_{BiBj}}{\partial\mathbf{b}^g} \cdot \delta\mathbf{b}^g + \frac{\partial\Delta\tilde{\mathbf{p}}_{BiBj}}{\partial\mathbf{b}^a} \cdot \delta\mathbf{b}^a
\end{aligned} \tag{5.50}$$

where the bias-dependent jacobians can be computed iteratively as well (see the appendix B. in [80]).

These final preintegrated measurements can now be integrated into a factor graph for optimization.

5.4.5 Visual-Inertial-Pressure Cost Function

We now detail the different error terms that we use to solve the proposed Visual-Inertial-Pressure SLAM. We recall that the related cost function is defined as:

$$\chi^* = \underset{\chi}{\arg\min}\left(E_{visual}(\chi) + E_{imu}(\chi) + E_{depth}(\chi)\right) \tag{5.51}$$

We first detail the IMU related error terms $E_{imu}(\chi)$, then the visual ones $E_{visual}(\chi)$ and finally the pressure related ones $E_{depth}(\chi)$.

5.4.5.1 IMU Error Terms

The IMU error terms to use within the NLLS optimization are constructed from the preintegrated measurements (Eq.(5.50)) :

$$\begin{aligned}
\mathbf{e}_{\Delta \mathbf{R}_{BiBj}} &= \hat{\mathbf{R}}_{BiBj} \boxminus \Delta\tilde{\mathbf{R}}_{BiBj} \\
&= \mathrm{Log}\left(\left(\tilde{\mathbf{R}}_{BiBj} \cdot \mathrm{Exp}\left(\frac{\partial \Delta\tilde{\mathbf{R}}_{BiBj}}{\partial \mathbf{b}^g} \cdot \delta \mathbf{b}^g\right)\right)^T \cdot \hat{\mathbf{R}}_{BiW} \cdot \hat{\mathbf{R}}_{WBj}\right)
\end{aligned} \tag{5.52}$$

$$\begin{aligned}
\mathbf{e}_{\Delta \mathbf{v}_{BiBj}} &= \hat{\mathbf{v}}_{BiBj} \boxminus \Delta\tilde{\mathbf{v}}_{BiBj} \\
&= \hat{\mathbf{R}}_{BiW} \cdot \left(\hat{\mathbf{v}}_{WBj} - \hat{\mathbf{v}}_{WBi} - \mathbf{g}_W \cdot \Delta t_{ij}\right) \\
&\quad - \left[\Delta\tilde{\mathbf{v}}_{BiBj} + \frac{\partial \Delta\tilde{\mathbf{v}}_{BiBj}}{\partial \mathbf{b}^g} \cdot \delta \mathbf{b}^g + \frac{\partial \Delta\tilde{\mathbf{v}}_{BiBj}}{\partial \mathbf{b}^a} \cdot \delta \mathbf{b}^a\right]
\end{aligned} \tag{5.53}$$

$$\begin{aligned}
\mathbf{e}_{\Delta \mathbf{p}_{BiBj}} &= \hat{\mathbf{p}}_{BiBj} \boxminus \Delta\tilde{\mathbf{p}}_{BiBj} \\
&= \hat{\mathbf{R}}_{BiW} \cdot \left(\hat{\mathbf{p}}_{WBj} - \hat{\mathbf{p}}_{WBi} - \hat{\mathbf{v}}_{WBi} \cdot \Delta t_{ij} - \frac{1}{2} \cdot \mathbf{g}_W \cdot \Delta t_{ij}^2\right) \\
&\quad - \left[\Delta\tilde{\mathbf{p}}_{BiBj} + \frac{\partial \Delta\tilde{\mathbf{p}}_{BiBj}}{\partial \mathbf{b}^g} \cdot \delta \mathbf{b}^g + \frac{\partial \Delta\tilde{\mathbf{p}}_{BiBj}}{\partial \mathbf{b}^a} \cdot \delta \mathbf{b}^a\right]
\end{aligned} \tag{5.54}$$

$$\begin{aligned}
\mathbf{e}_{\Delta \mathbf{b}^g_{BiBj}} &= \hat{\mathbf{b}}^g_{Bj} \boxminus \hat{\mathbf{b}}^g_{Bi} = \hat{\mathbf{b}}^g_{Bj} - \hat{\mathbf{b}}^g_{Bi} \\
\mathbf{e}_{\Delta \mathbf{b}^a_{BiBj}} &= \hat{\mathbf{b}}^a_{Bj} \boxminus \hat{\mathbf{b}}^a_{Bi} = \hat{\mathbf{b}}^a_{Bj} - \hat{\mathbf{b}}^a_{Bi}
\end{aligned} \tag{5.55}$$

Concatenating these error terms we model the IMU error term $E_{imu}(\chi)$ of Eq.(5.27):

$$E_{imu}(\chi) = \sum_{\mathfrak{K}^*} \left(\mathbf{e}_{imu}(\mathbf{X}_i, \mathbf{X}_j)^T \cdot {\boldsymbol{\Sigma}^{imu}_{BiBj}}^{-1} \cdot \mathbf{e}_{imu}(\mathbf{X}_i, \mathbf{X}_j)\right) \tag{5.56}$$

with the IMU error term $e_{imu}(\mathbf{X}_i, \mathbf{X}_j)$:

$$\mathbf{e}_{imu}(\mathbf{X}_i, \mathbf{X}_j) = \begin{bmatrix} \mathbf{e}_{\Delta \mathbf{R}_{BiBj}} & \mathbf{e}_{\Delta \mathbf{p}_{BiBj}} & \mathbf{e}_{\Delta \mathbf{v}_{BiBj}} & \mathbf{e}_{\Delta \mathbf{b}^g_{BiBj}} & \mathbf{e}_{\Delta \mathbf{b}^a_{BiBj}} \end{bmatrix}^T \tag{5.57}$$

and the related covariance $\boldsymbol{\Sigma}^{imu}_{BiBj}$ expressed as:

$$\boldsymbol{\Sigma}^{imu}_{BiBj} = \begin{bmatrix} \boldsymbol{\Sigma}^{\Delta \mathbf{I}}_{BiBj} & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & \boldsymbol{\Sigma}^{\mathbf{b}^{g,a}}_{BiBj} \end{bmatrix} \quad , \quad \boldsymbol{\Sigma}^{\mathbf{b}^{g,a}}_{BiBj} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathbf{b}^g} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\Sigma}^{\mathbf{b}^a} \end{bmatrix} \cdot \Delta t_{ij} \tag{5.58}$$

where $\mathbf{\Sigma}^{\Delta \mathbf{I}}_{BiBj}$ is extracted from Eq.(5.47) and $\mathbf{\Sigma}^{\mathbf{b}^{g,a}}_{BiBj}$ is set according to the IMU calibration.

5.4.5.2 Visual Error Terms

As we now work in the body frame, instead of the camera frame, we need to re-define the visual error terms $\mathbf{e}_{ij}$ that represent the reprojection error between the visual measurement $\tilde{\mathbf{x}}_{ij}$ and the correspond 3D landmark ${}_W\mathbf{l}_j$ in the keyframe $\mathbf{X}_i$. Assuming a known extrinsic calibration between the camera and the IMU, we can define the projection function $h(\cdot) : \mathbb{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^3$ that maps a 3D vector from the World frame to the camera frame as:

$$h(\mathbf{X}_i, {}_W\mathbf{l}_j) = \mathbf{R}_{CB} \cdot \left(\mathbf{R}_{BiW} \cdot {}_W\mathbf{l}_j + \mathbf{p}_{BiW}\right) + \mathbf{p}_{CB} \tag{5.59}$$

where $\mathbf{R}_{CB} \in \mathbb{SO}(3)$ and $\mathbf{p}_{CB} \in \mathbb{R}(3)$ are respectively the known rotation and translation between the camera and the IMU.

The resulting projection can then be used to define visual reprojection error terms $\mathbf{e}_{ij}$ from the known camera intrinsic calibration matrix $\mathbf{K}$, in the same way as in the pure Visual SLAM method (see Eq.(4.4) in page 70):

$$\mathbf{e}_{ij} = \tilde{\mathbf{x}}_{ij} - \pi\left(\mathbf{K} \cdot h(\mathbf{X}_i, {}_W\mathbf{l}_j)\right) \tag{5.60}$$

where $\pi(\cdot) : \mathbb{R}^3 \mapsto \mathbb{R}^2$ is defined as:

$$\pi\left(\mathbf{K} \cdot \mathbf{x}'\right) = \begin{bmatrix} \frac{x'}{z'} \cdot f_x + c_x \\ \frac{y'}{z'} \cdot f_y + c_y \end{bmatrix} \text{ with } \mathbf{x}' = h(\mathbf{X}_i, {}_W\mathbf{l}_j) \tag{5.61}$$

$$\text{and } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.62}$$

The visual energy term $E_{visual}(\chi)$ of Eq.(5.27) is finally defined as:

$$E_{visual}(\chi) = \sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \rho\left(\mathbf{e}_{ij}^T \cdot \mathbf{\Sigma}_{visual}^{-1} \cdot \mathbf{e}_{ij}\right) \tag{5.63}$$

where $\rho(\cdot)$ is the robust Huber norm.

5.4.5.3 Pressure Error Terms

The IMU sensor provides a way of sensing the gravity force direction and thus to align the World reference frame with the gravity axis. Thanks to this alignment and assuming that the gravity vector is collinear with the depth axis, the misalignment effect between the camera and the pressure sensor described in the previous sections (sec. 5.3.2.1) does not exist anymore.

Therefore, we can use the depth measurements as absolute measurements. In this Visual-Inertial-Pressure SLAM method, the depth energy term $E_{depth}\left(\chi\right)$ of Eq.(5.27) is defined as in strategy 1 of the Visual-Pressure SLAM (see section 5.3.3.1):

$$E_{depth}\left(\chi\right) = \sum_{i\in\mathfrak{K}} E_{depth}\left(\mathbf{X}_i\right) = \sum_{i\in\mathfrak{K}} \|\tilde{d}_i - \hat{p^z}_{WBi}\|^2_{\sigma^2_{depth}} \tag{5.64}$$

5.4.6 Visual-Inertial-Pressure Initialization

Before exploiting the IMU measurements for localization purpose, an initialization procedure has to be performed. In fact, we have to estimate the unknown parameters required to exploit the accelerometer and gyroscope readings within the Visual SLAM framework. These unknown parameters are: the direction of the gravity vector $\mathbf{g}_{WB_0}$ in the initial body frame, the initial velocities $\mathbf{v}_{WBi}$, the IMU biases $\mathbf{b}_i^g$ and $\mathbf{b}_i^a$ and the scale factor s relating the unscaled visual pose estimates to a metric scale.

Here we follow the initialization procedure proposed in [183] and refer the interested reader to their work for more details.

Using the estimated gravity vector $\mathbf{g}_{C_0}$, we compute the rotation matrix $\mathbf{R}_{WC_0}$ by aligning this gravity vector with the gravity in the world referential $\mathbf{g}_W = \left[0, 0, \mathrm{g}\right]$, where g is the gravitational constant:

$$\mathbf{g}_W = \mathbf{R}_{WC_0} \cdot \mathbf{g}_{C_0} \tag{5.65}$$

The metric scale factor s and the World alignment rotation matrix $\mathbf{R}_{WC_0}$ are then used to scale and align the frame states $\left(\mathbf{R}_{WB_k}, \mathbf{p}_{WB_k}, \mathbf{v}_{WB_k}\right)$. From this Visual-Inertial initialization, we hence expect to get a rough estimate of the scale factor s. The scale will then converge to its true value when solving the NLLS problem, as the estimation of the accelerometer's bias will refine it. Furthermore, the pressure sensor measurements will highly help in correcting the accelerometer's bias and the scale.

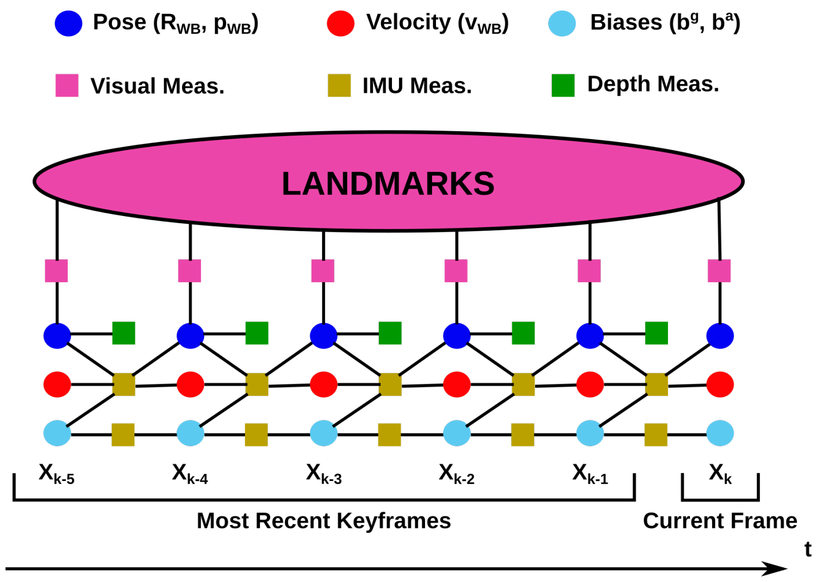


FIGURE 5.6: Factor graph related to the Visual-Inertial-Pressure SLAM problem. The states of every (key)frame is composed of a pose (position and orientation), a velocity and IMU related bias values.

5.4.7 Visual-Inertial-Pressure Algorithm

Once initialized, the Visual-Inertial-Pressure SLAM problem is solved in a fixed-lag smoothing fashion by performing optimization over the factor graph displayed in Fig. 5.6.

A fixed sized window of the N most recent keyframes along with the most recent frame is processed at every new frame. Moreover, all the 3D landmarks observed by at least two frames within this window are also added to the states to optimize. The state vector χ_i at each new frame is hence defined as:

$$\chi_i = \begin{bmatrix} \mathbf{X}_i & \chi_{KFi} & \chi_{\lambda i} \end{bmatrix} \tag{5.66}$$

where $\mathbf{X}_i$ is the state (Eq.(5.26)) related to the current frame, χ_{KFi} is the set of states related to the N most recent keyframes and $\chi_{\lambda i}$ is the set of 3D landmarks observed by at least two frames within χ_i.

The factor graph describing the problem to solve is illustrated in Fig. 5.6. The underlying NLLS formulation is hence defined following Eq.(5.27):

$$\chi^* = \underset{\chi}{\arg\min} \left(\sum_{i \in \mathfrak{K}} \sum_{j \in \mathfrak{L}_i} \rho \left(\|\mathbf{e}_{ij}\|^2_{\Sigma_{visual}} \right) + \sum_{\mathfrak{K}^*} \|\mathbf{e}_{imu}(\mathbf{X}_i, \mathbf{X}_j)\|^2_{\Sigma^{imu}_{B_iB_j}} \right. \tag{5.67}$$

$$\left. + \sum_{i \in \mathfrak{K}} \|\tilde{d}_i - \hat{p^z}_{WBi}\|^2_{\sigma^2_{depth}} \right) \tag{5.68}$$

This optimization problem is solved on-manifold with the Levenberg-Marquardt algorithm, in the same manner as in the Visual SLAM Bundle Adjustment (sec. 4.6). In order to fix the gauge of the problem and given that roll and pitch are observable from the IMU measurements, we only fix the position and yaw of the oldest keyframe in the window.

If a new keyframe is required by the SLAM algorithm, the current frame is turned into a keyframe and the oldest keyframe is removed from the window. Otherwise, the current frame is removed but its related IMU preintegrated measurements are kept and updated with the new coming IMU measurements, forming a new preintegrated factor for the next frame.

As the optimization run-time required to solve Eq.(5.68) might be longer than the delay between the acquisition of two consecutive images, the pose estimation at frame-rate is done by fixing the most recent keyframe along with the 3D landmarks (older keyframes are not included). The IMU biases are also fixed in this case. This step hence estimates only the current frame's pose and velocity using not only the visual measurements but also the IMU's and depth's ones. The resulting optimization problem can be solved in a very fast way as it does not involve many states.

5.4.8 Experimental Results

We now present the results obtained with this Visual-Inertial-Pressure SLAM method. We use the same data sequences than for the Visual-Pressure SLAM here (see section 5.3.5).

We compare the results obtained with the Visual-Inertial-Pressure algorithm to those obtains with the Visual-Pressure algorithm using strategy 2 and gauge relaxation. For clarity, we will refer to the Visual-Inertial-Pressure SLAM as UW-VIP and to the Visual-Pressure SLAM as UW-VP.

Table 5.3 and Table 5.4 display the absolute trajectory errors obtained on

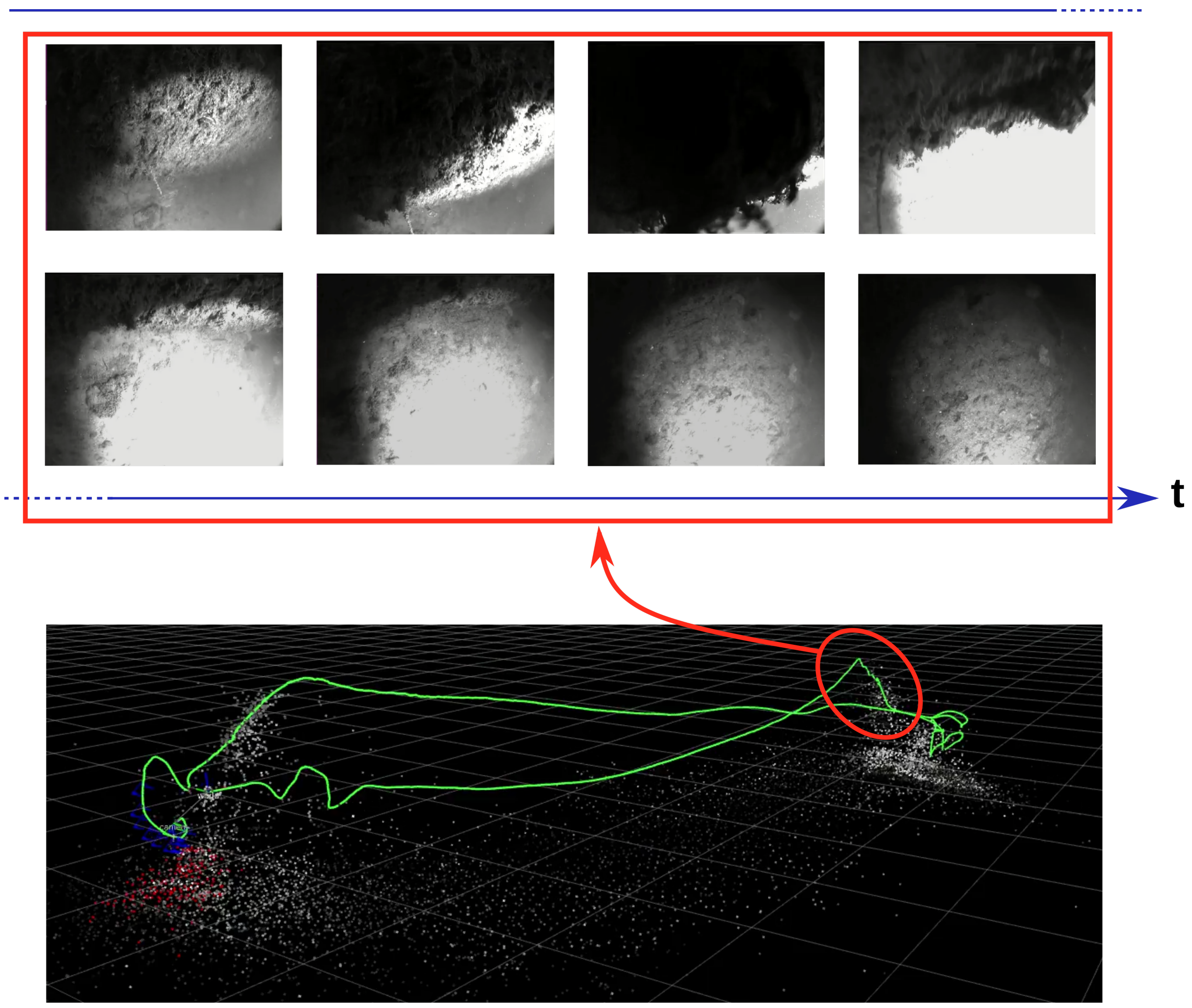


FIGURE 5.7: Estimated trajectory from UW-VIP on the sequence #7 from the harbor dataset. The red circle denotes the part of the sequence with a loss of visual information. Image samples from this part are given in the red box (to be read left to right and top to bottom).

each dataset. In general, there is no significant improvement over the accuracy of the estimated trajectories, which were already very accurate with UW-VP. However, we can see a significant improvement in robustness with UW-VIP. Indeed, it is able to handle the sequences #4 and #7 of the first dataset, whereas UW-VP failed because of short visual information loss. The final error is over 1 meter on these sequences but this is mainly due to the fact that, because vision could not be used during a short amount of time, the trajectory slightly drifted from the frame of reference, leading to an increase in terms of absolute trajectory error.

TABLE 5.3: Absolute trajectory errors (RMSE in m) on the Harbor dataset.

		Absolute Trajectory Error (m)	
Seq.	Length (m)	UW-VP	UW-VIP
# 1	39.3	0.49	0.42
# 2	75.6	0.36	0.37
# 3	23.6	0.25	0.26
# 4	55.8	X	1.56
# 5	28.5	0.13	0.09
# 6	19.5	0.04	0.06
# 7	32.9	X	1.16

TABLE 5.4: Absolute trajectory errors (RMSE in m) on the Archeological dataset.

		Absolute Trajectory Error (m)	
Seq.	Length (m)	UW-VP	UW-VIP
# 8	41.2	0.34	0.36
# 9	65.4	0.72	0.69

The trajectory estimated by UW-VIP on the sequence # 7 along with a sample of the images received during the short loss of visual information is illustrated in Fig. 5.7.

What happens in these cases of short losses of vision is the following. When the images acquired by the camera become unusable, the IMU and pressure sensor are providing the motion information required to continue the estimation of the current pose. However, the IMU is going to quickly drift during this time because of its evolving bias. This drift is a little bit limited by the depth measurements but only the visual measurements provide sufficient information for correctly estimating the biases of the IMU. As soon as vision is recovered, the biases of the IMU are corrected. If the timelapse without visual measurements is not too long, the drift will be recoverable and the biases correction will be propagated through the sliding window (Fig. 5.6). However, if vision is lost for more than a few seconds, the drift of the MEMS-IMU will be too important to be recoverable.

5.5 Conclusion

In this chapter, we have first investigated different strategies to estimate accurate and scaled localization from a monocular camera and a pressure sensor. The proposed Visual-Pressure SLAM is able to produce accurate and scaled trajectories by tightly coupling the visual and pressure measurements. We note here that some of the investigated strategies were based on the data sequences we had to test the algorithm. As the trajectories performed by the ROVs during these sequences were quite random, we had to deal with some issues such as low depth variations during the initialization phase. In a real scenario, with a *human in the loop*, we could imagine that vertical motions are applied on the ROV at the beginning to ensure a good initialization for instance. Yet, having a robust algorithm that can handle non ideal or predefined motions is always better.

An extension of this Visual-Pressure SLAM was then proposed to integrate a low-cost MEMS-IMU. We showed that the final Visual-Inertial-Pressure SLAM method is more robust and can handle short loss of visual information. This Visual-Inertial-Pressure SLAM runs in real-time thanks to the use of preintegration for the IMU measurements.

The accuracy of the Visual-Inertial-Pressure estimations could probably be improved by means of marginalization [138], as a way of keeping information about keyframes and landmarks that leave the optimization window. However, marginalization is still tricky to use optimally without leading to an excessive computational overload [109]. Besides, even if adding the measurements of a pressure sensor and a low-cost MEMS-IMU helped in improving the localization and the robustness of the SLAM algorithm, drift might still occur. This especially true when vision is really poor. The only way of recovering from such drift would be to add a loop-detection feature to the SLAM algorithm. These suggestions are left for future work.

Chapter 6

Field Experiments and Online Photogrammetry

Contents

6.1 Introduction

In order to validate the SLAM methods presented in the previous chapters, we have designed two acquisition systems suited to our needs. These acquisition systems are self-contained in that they both embed the required sensors and a computer to run the algorithms in real-time. Their design hence makes them easy to embed on almost any ROV and they have been successfully tested in real-case scenarios during an archaeological camping conducted by the DRASSM (Department of underwater archaeology, French Ministry of Culture).

We have taken the opportunity offered by this campaign to further record a large dataset. It is this dataset that has been used in the previous chapter to evaluate the performance of the proposed SLAM methods. We have named this dataset AQUALOC [74, 72] and we have publicly released it for the benefit of the community.

The first topic of this chapter is about the presentation of these acquisition systems and of the AQUALOC dataset. Moreover, the data sequences contained in AQUALOC have given us material to work on for the development of a 3D reconstruction method. This chapter further presents a new dense 3D reconstruction method, compliant with a monocular setup. The proposed 3D reconstruction method takes as input both the images acquired by the camera and the output of the SLAM methods proposed in this thesis. The 3D reconstruction module runs online and can therefore be seen as a way of performing online photogrammetry.

This chapter is organized as follows. We first describe the acquisition systems tested during our field experiments. We also deeply describe the AQUALOC dataset and the acquisition conditions. Last, we present the proposed 3D reconstruction method suited to monocular systems.

6.2 Dataset Acquisition

In order to run the SLAM algorithm online during ROVs operational surveys, we have desgined acquisition systems that are composed of a monochromatic camera, a Micro Electro-Mechanical System (MEMS) based Inertial Measurement Unit (IMU) and a pressure sensor. These acquisition systems further embed a computing unit for running in real-time the SLAM algorithm.

These acquisition systems have been embedded on ROVs equipped with lighting systems and navigating close to the seabed in different environments and at different depths. These experiments have allowed us to validate the use of the SLAM algorithm online and to record a dataset with synchronized measurements.

The recorded video sequences exhibit the typical visual degradation induced by the underwater environment such as turbidity, backscattering, shadows and strong illumination shifts caused by the artificial lighting systems.

Three different sites have been explored to create a dataset and test the SLAM algorithm: a harbor and two archaeological sites. The recording of the sequences occurred at different depths, going from a few meters, for the harbor, to several hundred meters, for the archaeological sites. The video sequences are hence highly diversified in terms of scenes (low-textured areas, texture repetitive areas...) and of scenarios (exploration, photogrammetric surveys, manipulations...).

In order to quantitatively evaluate SLAM algorithms, we have computed a comparative baseline to evaluate the quality and accuracy of the SLAM estimated trajectories. As the acquisition of ground truth is very difficult in natural underwater environments, we have used the state-of-the-art Structure-from-Motion (SfM) library Colmap [197] to compute comparative baseline trajectories for each sequence. Colmap processes offline the sequences and performs a 3D reconstruction to estimate the positions of the camera. This 3D reconstruction is done by matching exhaustively all the images composing a sequence, which allows the detection of many loop-closures and, hence, the computation of accurate trajectories, assessed by low average reprojection errors (< 1 pixel on all the sequences).

In addition to using this dataset for evaluating the efficiency of the proposed SLAM algorithm, we have publicly released this dataset [74, 72] in order to give to the community an opportunity to work on data that are difficult to acquire. Indeed, the logistic (ship availability) and the required equipment (deep-sea compliant underwater vehicles and sensors), as well as regulations (official authorizations), can prevent possible works on this topic. We hope that the availability of this dataset will increase the development of algorithms dedicated to the underwater environment. Both raw and ROS bag formatted field data are provided along with the full calibration of the sensors (camera and IMU). Moreover, the provided comparative baseline makes this dataset suitable for benchmarking Visual SLAM and Visual-Inertial(-Pressure) SLAM algorithms.

The rest of this section is organized as follows. First, we review some works that have led to the release of similar datasets. Then, we deeply detail the design of the acquisition systems used and the calibration procedures employed. Next, we present the acquired dataset and the acquisition conditions on each site are detailed, highlighting the associated challenges for visual localization. Finally, the processing of the data sequences to create a baseline is described.

6.2.1 Related Work

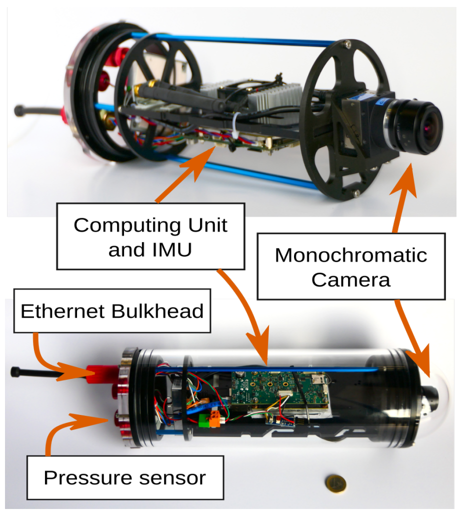
In ground and aerial robotics, the availability of many public datasets, such as KITTI [94], Malaga [19] or EuRoC [24], to cite a few, has greatly impacted the development of VSLAM methods.

In the underwater field, there is a limited amount of public datasets dedicated to the evaluation of SLAM methods. Moreover, the fact that these data are difficult to acquire, because of the required equipment and logistic, limits the development of new methods. In [17], a dataset containing the measurements of navigational sensors , stereo cameras and a multi-beam sonar was proposed. Another another dataset dedicated to localization and mapping in an underwater cave from sonar measurements was proposed in [154]. Images acquired by a monocular camera are also given for the detection of cones precisely placed in order to have a mean of estimating drift. However, in both datasets, the acquisition rate of the cameras is too low (<10 Hz) for most of VSLAM methods. A synthetic dataset was created in [54], simulating the navigation of a vehicle in an underwater environment and containing monocular cameras measurements at a frame-rate of 10 Hz. Many public datasets have also been made available by the Oceanography community through national websites (`https://www.data.gov/`, `http://www.marine-geo.org`). However, these datasets have not been gathered with the aim of providing data suitable for VSLAM and often lack essential information such as the calibration of their sensors setups.

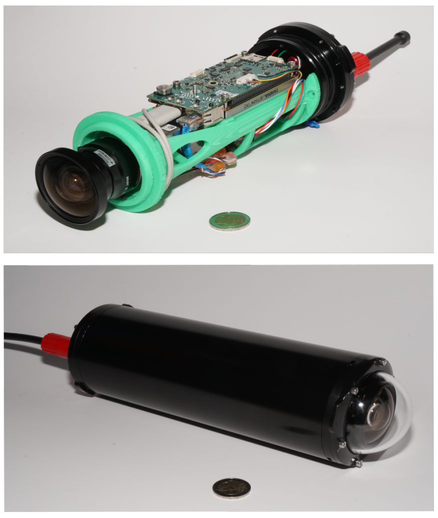
With respect to these works, we propose a new dataset aiming at the development of VSLAM and Visual-Inertial(-Pressure) SLAM methods dedicated to the underwater environment.

6.2.2 The Acquisition Systems

In order to acquire the sequences of the dataset, we have designed two similar underwater systems. These acquisition systems have been designed to allow the localization of underwater vehicles from a minimal set of sensors in order to be as cheap and as versatile as possible. Both systems are equipped with a monochromatic camera, a pressure sensor, a low-cost MEMS-IMU and an embedded computer. The camera is placed behind an acrylic dome to minimize the distortion effects induced by the difference between water and air refractive indices. The image acquisition rate is 20 Hz. The IMU delivers measurements from a 3-axes accelerometer, 3-axes gyroscope and



(A) System A



(B) System B

FIGURE 6.1: The acquisition systems equipped with a monocular monochromatic camera, a pressure sensor, an IMU and a computer along with the sensors' reference frames.

3-axes magnetometer at 200 Hz. The embedded computer is a Jetson TX2 running Ubuntu 16.04 and is used to record synchronously the sensors' measurements thanks to the ROS middleware. The Jetson TX2 is equipped with a carrier board embedding the mentioned MEMS-IMU and a 1 To NVME SSD to directly store the sensors measurements, thus avoiding any bandwidth or package loss issue. An advantage of the self-contained systems that we have developed, is that they are independent of any robotic architecture and can thus be embedded on any kind of Remotely Operated Vehicle (ROV) or Autonomous Underwater Vehicle (AUV). The interface can either be Ethernet or a serial link, depending on the host vehicle's features.

To record data at different depths, we have designed two systems that we will refer to as "System A" and "System B". These systems have the same overall architecture, but they differ on the camera model, the pressure sensor type and the diameter and material of the enclosure. System A (Fig. 6.1a) is designed for shallow waters and was used to acquire the sequences in the harbor. Its camera has been equipped with a wide-angle lens, which can be modeled by the fisheye distortion model. The pressure sensor is rated for 30 bars and delivers depth measurements at a maximum rate of 10 Hz. System A is protected by an acrylic enclosure, rated for a depth of 100 meters. System B (Fig. 6.1b) was used to record the sequences on the archaeological sites at larger depths. Its camera has a slightly lower field of view and the lens can be modeled by the radial-tangential distortion model. It embeds a pressure sensor rated for 100 bars, delivering depth measurements at 60

System A (Harbor sequences)	**Camera sensor**	**UEye - UI-1240SE**
	Resolution	640×512 px
	Sensor	Monochromatic
	Frames per second	20 fps
	Lens	**Kowa LM4NCL C-Mount**
	Focal length	3.5mm
	Pressure Sensor	**MS5837 - 30BA**
	Depth range	0 - 290m
	Resolution	0.2 mbar
	Output frequency	5-10 Hz
	Inertial Measurement Unit	**MEMS - MPU-9250**
	Gyroscope frequency	200 Hz
	Accelerometer frequency	200 Hz
	Magnetometer frequency	200 Hz
	Embedded Computer	**Nvidia - Tegra Jetson TX2**
	Carrier board	Auvidea J120 - IMU
	Storage	NVME SSD 1 To
	Housing	**4" Blue Robotics Enclosure**
	Enclosure	33.4 x 11.4 cm
	Enclosure Material	Acrylic
	Dome	4" Blue Robotics Dome End Cap
System B (Archaeo. sequences)	**Camera sensor**	**UEye - UI-3260CP**
	Resolution	968×608 px
	Sensor	Monochromatic
	Frames per second	20 fps
	Lens	**Kowa LM6NCH C-Mount**
	Focal length	6mm
	Pressure Sensor	**Keller 7LD - 100BA**
	Depth range	0 - 990m
	Resolution	3 mbar
	Output frequency	60 Hz
	Inertial Measurement Unit	***Same as System A***
	Embedded Computer	***Same as System A***
	Housing	**3" Blue Robotics Enclosure**
	Enclosure	25.8 x 8.9 cm
	Enclosure Material	Aluminium
	Dome	3" Blue Robotics Dome End Cap

TABLE 6.1: Technical details about the acquisition systems A and B.

Hz. Its enclosure is made of aluminum and is 400 meters depth rated. The technical details about both systems and their embedded sensors are given in table 6.1.

Each camera-IMU setup has been cautiously calibrated to provide the intrinsic and extrinsic parameters required to use it for localization purpose. We have used the toolbox Kalibr [83, 84] along with an apriltag target to compute all the calibration parameters.

The cameras calibration step allows to obtain an estimate of the focal lengths, principal points and distortion coefficients. These parameters can then be used to undistort the captured images and to model the image formation pipeline, with the following notation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \Pi_{\mathbf{K}} \left(\mathbf{R}_{\mathrm{w}}^{\mathrm{cam}} \mathbf{X}_{\mathrm{w}} + \mathbf{t}_{\mathrm{w}}^{\mathrm{cam}} \right) \tag{6.1}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x . \frac{x_{\mathrm{cam}}}{z_{\mathrm{cam}}} + c_x \\ f_y . \frac{y_{\mathrm{cam}}}{z_{\mathrm{cam}}} + c_y \end{bmatrix} = \Pi_{\mathbf{K}} \left(\mathbf{X}_{\mathrm{cam}} \right) \tag{6.2}$$

$$\text{with } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{X}_{\mathrm{cam}} = \begin{bmatrix} x_{\mathrm{cam}} \\ y_{\mathrm{cam}} \\ z_{\mathrm{cam}} \end{bmatrix}$$

where $\Pi_{\mathbf{K}}(\cdot)$ denotes the projection: $\mathbb{R}^3 \mapsto \mathbb{R}^2$, $\mathbf{K}$ is the calibration matrix, $\mathbf{X}_{\mathrm{w}} \in \mathbb{R}^3$ is the position of a 3D landmark in the world frame, $\mathbf{R}_{\mathrm{w}}^{\mathrm{cam}} \in SO(3)$ and $\mathbf{t}_{\mathrm{w}}^{\mathrm{cam}} \in \mathbb{R}^3$ denote the rotational and translational components of the transformation from the world frame to the camera frame, $\mathbf{X}_{\mathrm{cam}} \in \mathbb{R}^3$ is the position of a 3D landmark in the camera frame, f_x and f_y denotes the focal lengths in pixels and (c_x, c_y) is the principal point of the camera (also in pixels). The distorted pixel coordinates (u, v) are the projection of $\mathbf{X}_{\mathrm{cam}}$ into the undistorted image frame.

As these parameters are medium dependent, the calibration has been performed in water to account for the additional distortion effects at the dome's level. The results of the calibration of the fisheye camera can be seen in figure 6.2.

The camera-IMU setup calibration allows to estimate the extrinsic parameters of the setup, that is the relative position of the camera with respect to the IMU, and the time delay between camera's and IMU's measurements. This relative position is expressed by a rotation matrix $\mathbf{R}_{\text{cam}}^{\text{imu}}$ and a translation vector $\mathbf{t}_{\text{cam}}^{\text{imu}}$. Camera and IMU's poses relate to each other through:

$$\mathbf{T}_{\text{cam}}^{\text{w}} = \mathbf{T}_{\text{imu}}^{\text{w}} \mathbf{T}_{\text{cam}}^{\text{imu}} \tag{6.3}$$

$$\text{with } \mathbf{T}_{\text{cam}}^{\text{imu}} \doteq \begin{bmatrix} \mathbf{R}_{\text{cam}}^{\text{imu}} & \mathbf{t}_{\text{cam}}^{\text{imu}} \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

$$\text{and } (\mathbf{T}_{\text{cam}}^{\text{w}})^{-1} = \mathbf{T}_{\text{w}}^{\text{cam}} \doteq \begin{bmatrix} \mathbf{R}_{\text{w}}^{\text{cam}} & \mathbf{t}_{\text{w}}^{\text{cam}} \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

where $\mathbf{R}_{\text{cam}}^{\text{imu}} \in SO(3)$, $\mathbf{t}_{\text{cam}}^{\text{imu}} \in \mathbb{R}^3$, $\mathbf{T}_{\text{cam}}^{\text{w}} \in SE(3)$, $\mathbf{T}_{\text{w}}^{\text{cam}} \in SE(3)$, $\mathbf{T}_{\text{cam}}^{\text{imu}} \in SE(3)$ and $\mathbf{T}_{\text{imu}}^{\text{w}} \in SE(3)$. $\mathbf{T}_{\text{cam}}^{\text{w}}$ and $\mathbf{T}_{\text{imu}}^{\text{w}}$ respectively represent the poses of the camera and of the body, with respect to the world frame. $\mathbf{T}_{\text{w}}^{\text{cam}}$ is the inverse transformation of $\mathbf{T}_{\text{cam}}^{\text{w}}$ and $\mathbf{T}_{\text{cam}}^{\text{imu}}$ is the transformation from the camera frame to the IMU frame.

Before estimating these extrinsic parameters, the IMU noise model parameters have been derived from an Allan standard deviation plot [58], obtained by recording the gyroscope and accelerometer measurements for several hours, while keeping the IMU still. These noise parameters are then fed into the calibration algorithms to model the IMU measurements. As these parameters (IMU noises, camera-IMU relative transformation and measurements' time delay) are independent of the medium (air or water), they have been estimated in air. Doing this calibration step in air allowed to perform easily the fast motions required to correlate the IMU measurements to the camera's ones.

All the calibration results are included in the dataset, that is the cameras' models (including the intrinsic parameters and the distortion coefficients), the IMUs' noise parameters, the relative transformation from the camera to the IMU and the time delay between the cameras' and the IMUs' measurements.

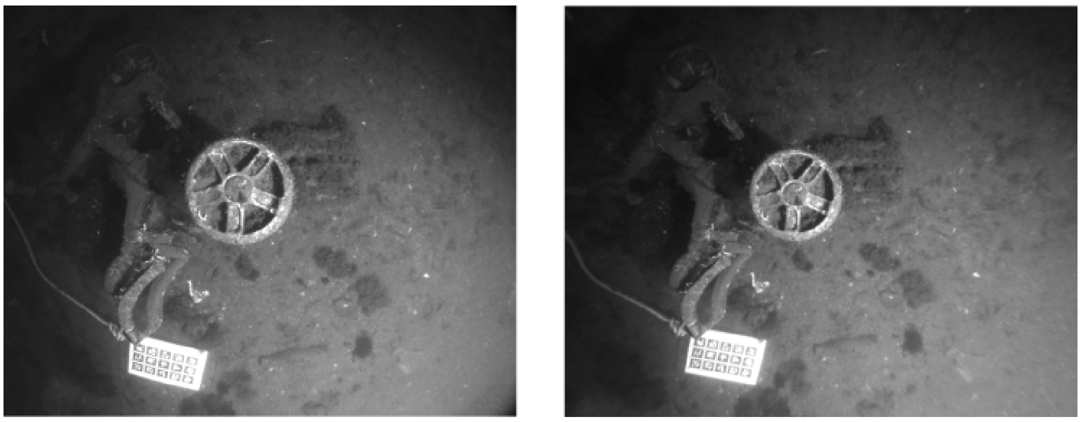


FIGURE 6.2: Distortion effects removal from Kalibr calibration on one of the harbor sequences. Left: raw image. Right: undistorted image.

6.2.3 The AQUALOC dataset

These systems have been used to record a dataset. As explained in section 6.2.2, System A was used to record the shallow harbor sequences, while System B was used on the two deep archaeological sites. The dataset consist in a total of 17 sequences: 7 recorded in the harbor, 4 on the first archaeological site and 6 on the second site. As each of these environments is in some ways different from the others, we describe the sequences recorded in each environment separately. Table 6.2 summarizes the specificities of each data sequence. Note that, for each sequence, the starting and ending points are approximately the same. In most of the sequences, there are closed loops along the performed trajectories. Some sequences also slightly overlap, which can be useful for the development of relocalization features.

6.2.3.1 Harbor sequences

To record the harbor sequences, System A was embedded on the lightweight ROV *Dumbo* (*DRASSM-LIRMM*) with the camera facing downward, as shown in figure 6.3. The ROV was navigating at a depth of 3 to 4 meters over an area of around 100 m^2. Although the sun illuminates this shallow environment, a lighting system was used in order to increase the signal-to-noise ratio of the images acquired by the camera. The explored area was mostly planar but the presence of several big objects made it a real 3D environment, with significant relief.

For each sequence, loops are performed and an apriltag calibration target is used as a marker for starting and ending points. On these sequences, vision

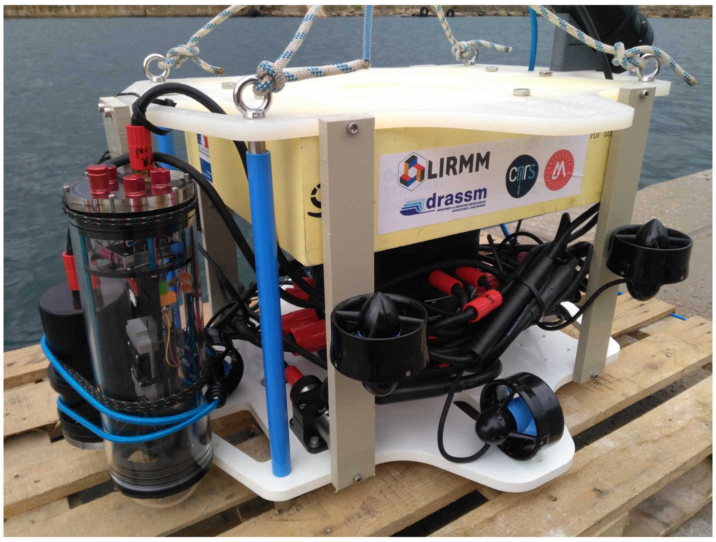


FIGURE 6.3: The Remotely Operated Vehicle *Dumbo* and the acquisition system A, used to record the harbor sequences.

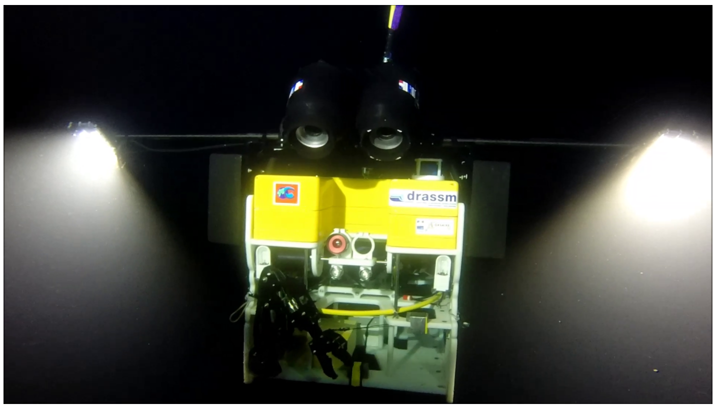


FIGURE 6.4: The Remotely Operated Vehicle *Perseo*, used on the archaeological sites.
Credit: F. Osada - DRASSM / Images Explorations.

is mostly degraded by turbidity, light absorption, strong illumination variations and backscattering. In two sequences, visual information even becomes unavailable for a few seconds because of collisions with surrounding objects. Another challenge is the presence of areas with seagrass moving because of the swell. Moreover, the ROV is sensitive to waves and tether disturbances, which results in roll and pitch variations.

6.2.3.2 Archaeological sites sequences

The archaeological sites sequences were recorded in the Mediterranean sea, off Corsica's coasts. The System B, designed for deep waters, was embedded on the *Perseo* ROV (*Copetech SM Company*) displayed in Fig. 6.4. In the way it was attached to the ROV, the camera viewing direction made a small

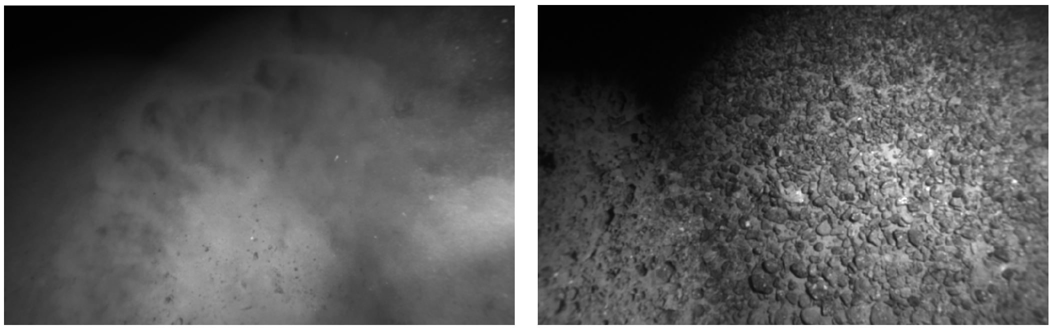
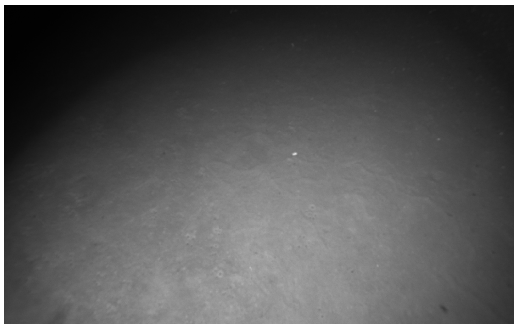

(A) Sandy cloud (B) Texture repetitive area (ballast stones)

FIGURE 6.5: Images acquired on the first archaeological site (depth: 270m).

angle with the vertical line ($\approx 20 - 30$ degrees). *Perseo* was equipped with two powerful LED lights (250,000 lumens each) and with two robotics arms for manipulation purposes. As localization while manipulating objects is a valuable information, to grab an artifact for instance, in some sequences the robotic arms are in the camera's field of view. A total of 10 sequences have been recorded on these sites, with 3 sequences taken on the first site and 7 on the second one.

The first archaeological site explored was located at a depth of approximately 270 meters and hosted the remains of an antic shipwreck. Hence, this site is mostly planar and presents mainly repetitive textures, due to numerous small rocks that were used as ballast in this antic ship (Fig. 6.5a). These sequences are affected by turbidity and moving sand particles, increasing backscattering and creating sandy clouds (Fig. 6.5b). These floating particles are stirred up from the seabed by the water flows of the ROV's thrusters and lead to challenging visual conditions. A shadow is also omnipresent in these sequences in the left corner of the recorded images, because of the orientation of the lighting system, optimized for the ROV's own camera and not for the SLAM ones.

The second visited archaeological site was located at a depth of approximately 380 meters. On this site a hill of amphorae is present (Fig. 6.6b), whose top is culminating a few meters above the surrounding seabed level. During these sequences, the ROV was mainly operated for manipulation and photogrammetry purposes. While the amphorae presented high texture, the ROV was also sometimes hovering over low-textured sandy areas around the hill of amphorae (Fig. 6.6a). Because of the presence of these amphorae, marine wildlife was present on this site. Hence, the environment is quite dynamic, with many fishes getting in the field of view of the camera and many


(A) Low texture area

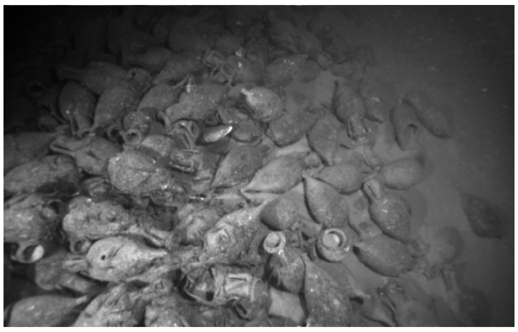

(B) Hill of amphorae

FIGURE 6.6: Images acquired on the second archaeological site (depth: 380m).

Site	Sequence	Duration	Length	Visual Disturbances					
				Turbidity	Collisions	Backscattering	Sandy clouds	Dynamics	Robotic Arm
Harbor (depth ≈ 4 m) Acquired by system A, embedded on a lightweight ROV	#01	3′49"	39.3m	X	-	X	-	-	-
	#02	6′47"	75.6m	X	-	X	-	-	-
	#03	4′17"	23.6m	X	-	X	-	-	-
	#04	3′26"	55.8m	X	X	X	-	-	-
	#05	2′52"	28.5m	X	-	X	-	-	-
	#06	2′06"	19.5m	X	-	X	-	-	-
	#07	1′53"	32.9m	X	X	X	-	-	-
First Archaeological Site (depth ≈ 270 m) Acquired by System B, embedded on a medium workclass ROV	#01	14′39"	32.4m	X	-	X	X	X	X
	#02	7′29"	64.3m	X	-	X	X	X	-
	#03	5′16"	10.7m	X	-	X	X	-	-
Second Archaeological Site (depth ≈ 380 m) Acquired by System B embedded on a medium workclass ROV	#04	11′09"	18.1m	X	-	X	X	X	X
	#05	3′19"	42.0m	X	-	X	-	X	-
	#06	2′49"	31.8m	X	-	X	-	X	-
	#07	9′29"	122.1m	X	-	X	-	X	-
	#08	7′49"	41.2m	X	-	X	-	X	-
	#09	5′49"	65.3m	X	-	X	-	X	-
	#10	11′54"	83.5m	X	-	X	-	X	-

TABLE 6.2: Details on all the AQUALOC sequences and their associated visual disturbances.

shrimps moving in the vicinity of the amphorae. In one of the sequences, both arms get in front of the camera. Otherwise, the visual degradation are the same as on the first site.

6.2.4 Comparative Baseline

As the acquisition of a ground truth is very difficult in natural underwater environment, we have used the state-of-the-art Structure-from-Motion (SfM) software Colmap [197] to offline compute a 3D reconstruction for each sequence and extract a reliable trajectory from it. By setting very low the features extraction parameters, we were able to extract enough SIFT features [146] to robustly match the images of each sequence. Performing a matching of the images in an exhaustive way, that is trying to match each image to all the other ones, allows to get a reliable trajectory reconstruction, as many closed loops can be found (Fig. 6.7). In Table 6.4, we provide statistics for each sequence about Colmap's 3D reconstructions to highlight the reliability

	Harbor sequences						
	#01	#02	#03	#04	#05	#06	#07
Nb. of used images	918	1590	1031	770	692	508	447
Nb. of 3D points	112659	305783	355130	194407	236845	188807	181964
Mean tracking length	14.9	13.2	17.2	9.7	10.7	12.1	9.5
Mean reproj. err. (px)	0.896	0.816	0.713	0.715	0.688	0.733	0.846

TABLE 6.3: Colmap trajectories reconstruction statistics on the Harbor sequences. The number of provided images, the number of reconstructed 3D points, the mean tracking length for the 3D points and the mean reprojection error for the 3D reconstruction are given for each sequence.

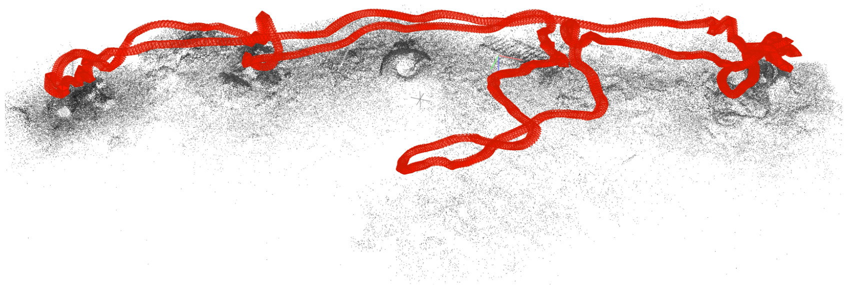
	Archeological sites sequences									
	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10
Nb. of used images	880	445	311	637	200	170	569	470	350	715
Nb. of 3D points	196857	174514	160531	249048	42877	45799	251620	237882	114814	329686
Mean tracking length	23.5	12.6	8.4	8.5	7.6	6.7	7.4	9.1	7.9	9.2
Mean reproj. err. (px)	0.746	0.621	0.474	0.673	0.601	0.569	0.645	0.616	0.660	0.661

TABLE 6.4: Colmap trajectories reconstruction statistics the Archaeological sites sequences. The number of provided images, the number of reconstructed 3D points, the mean tracking length for the 3D points and the mean reprojection error for the 3D reconstruction are given for each sequence.

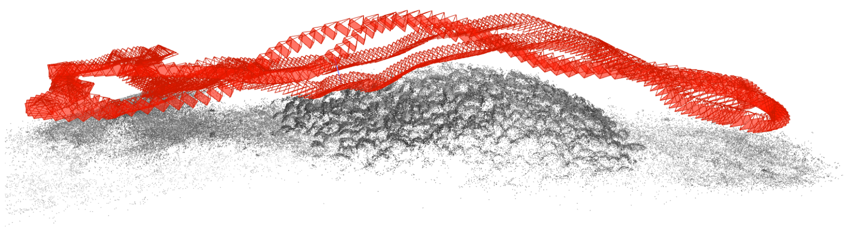
of the reconstructed models. These statistics include the number of images used, the number of estimated 3D points, the average track length of each 3D points (*i.e.* the number of images observing a given 3D point) and the average reprojection error. The high average track lengths for each sequence (going from 6.7 to more than 20) assess the accuracy of the 3D points' estimation, as it leads to a high redundancy in the bundle adjustment steps of the reconstruction. Moreover, given these high track lengths, the average reprojection error is a good indicator of the overall quality of a SfM 3D model and for each one of the sequences this error is below 0.9 pixel.

The extracted trajectories have been scaled using the pressure sensor measurements and hence provide metric positions. Although these trajectories cannot be considered as being perfect ground truths, we believe that it provides a fair baseline to evaluate and compare online localization methods. Evaluation of such methods can be done using the standard Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) metrics [212].

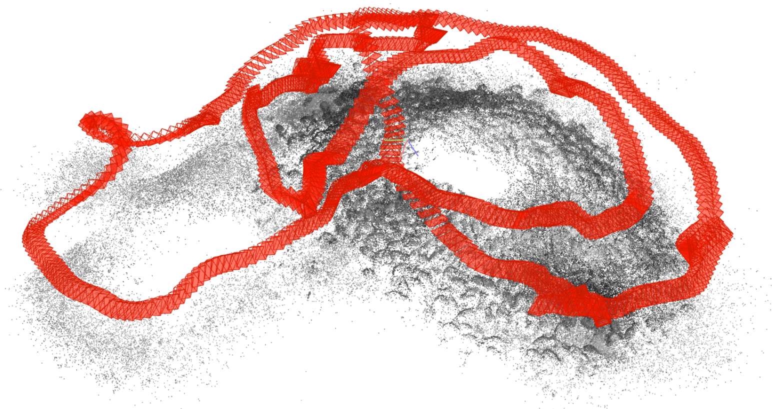
Furthermore, we have made available the list of overlapping images (*i.e.* matching) according to Colmap for each sequence. These files could hence be used to evaluate the efficiency of loop-closure or image retrieval methods.



(A) Colmap reconstruction - Harbor #02



(B) Colmap reconstruction - Archaeological Site #07



(C) Colmap reconstruction - Archaeological Site #10

FIGURE 6.7: Examples of trajectories reconstructed with Colmap. The camera's poses are represented in red and the reconstructed 3D landmarks in black.

6.2.5 Discussion

The presented datasets and acquisition systems have provided the data used in the chapter 5 to quantitatively evaluate the accuracy of the proposed SLAM algorithm. Furthermore, we have been able to run the SLAM algorithm in real-time on the Tegra TX2 embedded within the acquisition systems during the surveys. These tests thus allowed us to validate the different algorithms developed during this thesis for the purpose of localization and mapping in an underwater archaeological context.

This dataset has also provided us material to work on for developing a dense 3D reconstruction method. By integrating a 3D reconstruction module to the SLAM algorithm, we wish to further enhance the usefulness of the proposed system for piloting and navigation purposes during archaeological surveys. This 3D reconstruction method is detailed in the next section.

6.3 Towards Online Photogrammetry

We now propose a method to perform dense 3D reconstruction online. This method is built upon the SLAM methods developed in this thesis and is applicable with the acquisition systems described in this chapter.

The SLAM algorithm provides a way to estimate the localization of an underwater vehicle along with a sparse 3D map composed of the 3D landmarks estimated for the visual localization purpose. The 3D reconstruction method we propose is adapted to the monocular case and leverages all the keyframes and 3D landmarks estimated by the SLAM algorithm.

This method assumes that once a keyframe is old enough to be fixed within the SLAM algorithm, its pose and observed 3D landmarks estimates have converged towards their true values and are hence accurate enough to be reliable. These 3D observations are augmented and used as depth seeds to create a depth map. These seeds are then interpolated using a 2D Delaunay triangulation. From this, we get 2D depth maps that we can turn into 3D dense point clouds using the known keyframe's pose and camera's calibration matrix. These 3D point clouds are finally used to create 3D meshes through a *Truncated Signed Distance Field* (TSDF) [48].

This 3D reconstruction module is run in an individual thread within the SLAM algorithm for the creation of the dense 3D reconstruction in parallel of the SLAM estimations. The reconstruction is hence performed online but, because of the monocular camera that prevents a direct observation of the 3D structure of the imaged scenes, there is a slight delay between the dense 3D reconstruction and the SLAM estimations.

6.3.1 Turning sparse 3D measurements into 2D depth maps

The 3D reconstruction module receive every keyframe leaving the optimization window of the SLAM algorithm. This way, the 3D reconstruction module only process keyframes accurately estimated. Every time a keyframe is received, we process it as follows:

6.3.1.1 Redundancy Check

First, we check the similarity of this keyframe with the last processed keyframe. This is done by counting the number of common 3D landmarks they both observe and comparing it to the total number of 3D landmarks observed by the current keyframe. If this number drops below a threshold ($\leq 75\%$), we consider the new keyframe informative enough to be processed. Otherwise, the received keyframe is dropped and the module waits for the next keyframe. This similarity check allows to reduce the computational time allocated to the 3D reconstruction thread by avoiding redundant computations.

6.3.1.2 3D Augmentation

Secondly, if the keyframe is selected for the dense 3D processing, a 3D augmentation step is performed using the current keyframe and the last keyframe used in this 3D reconstruction pipeline. This 3D augmentation step aims at increasing the number of 3D observations (*i.e.* keypoints corresponding to known 3D landmarks) per keyframe.

This 3D augmentation is done as follows. First, many new keypoints are detected in the last keyframe in order to get a dense spread of keypoints over the image. In practice, between 1000 and 2000 points are sought in this step and we try to detect at least one keypoint every 5 pixels. Then, all the keypoints belonging to this previous keyframes and not observed in the current keyframe are tracked in the current keyframe's image through optical flow tracking with the KLT method. For the keypoints that already correspond to a 3D landmark, a prior solution is provided to the KLT tracking by projecting the corresponding landmarks into the current keyframe.

The keypoints that are successfully tracked are then run through an outlier test using the epipolar geometry. Using the known poses of both keyframes, we compute the fundamental matrix $\mathbf{F}$ defined by their relative configuration:

$$\mathbf{F} = \left(\mathbf{K} \cdot \mathbf{\Delta R}^T\right)^{-T} \cdot (\mathbf{K} \cdot \mathbf{\Delta R} \cdot \mathbf{\Delta t})^{\wedge} \cdot \mathbf{K}^{-1} \tag{6.4}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera's calibration matrix, $\mathbf{\Delta R} \in \mathbb{SO}(3)$ is the relative rotation matrix from the current keyframe to the last keyframe, $\mathbf{\Delta t} \in \mathbb{R}^3$ is the relative translation from current keyframe to last keyframe and $(\cdot)^{\wedge}$ is the skew-symmetric matrix operator.

The outliers removal is then done by computing the geometrical error of each keypoint tracked with respect to the keyframe's epipolar geometry. The fundamental matrix **F** defines the epipolar planes between two camera's images. These planes are constructed from corresponding lines in both images and we know from epipolar geometry that a 3D point that projects onto such a line in one image has to project on the corresponding line in the other image. The fundamental matrix **F** encodes this relation as follows:

$$\mathbf{x}^T \cdot \mathbf{F} \cdot \mathbf{x}' = 0 \tag{6.5}$$

with $\mathbf{x}' \in \mathbb{R}^3$ and $\mathbf{x} \in \mathbb{R}^3$ are the homogeneous coordinates of matching keypoints between the last keyframe and the current keyframe, respectively.

Because of noise in the visual measurements, Eq.(6.5) is never perfectly satisfied. The geometrical error r of each tracked keypoint is then computed as the pixel distance from the corresponding epipolar line:

$$\mathbf{l} = \mathbf{F} \cdot \mathbf{x}' = \begin{bmatrix} l_x & l_y & l_z \end{bmatrix}^T \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T \tag{6.6}$$

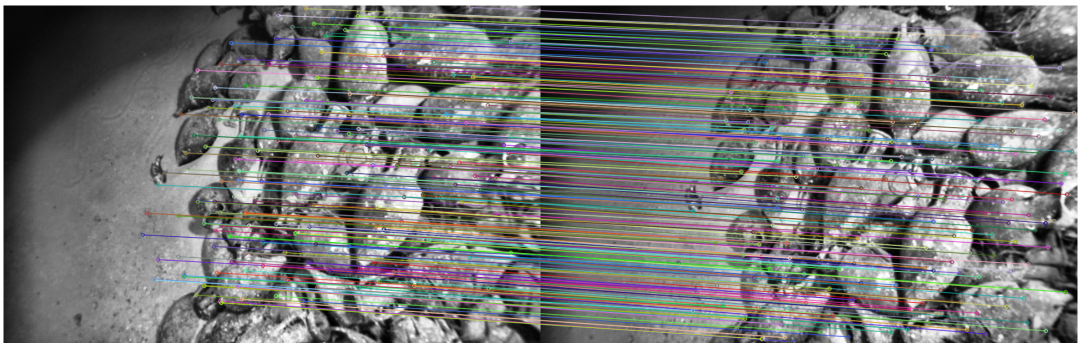
$$r = \frac{l_x \cdot u + l_y \cdot v + l_z}{\sqrt{l_x^2 + l_y^2}} \tag{6.7}$$

Each tracked keypoint whose epipolar error r is higher than 1.5 pixels is discarded. The keypoints that pass the epipolar check are then triangulated and, those with a reprojection error lower than 1 pixel in both images, are added to the set of 3D keypoints of both keyframes. We voluntarily prune many keypoints in this step in order to prevent the inclusion of spurious matchings. Results of this 3D augmentation process are illustrated in Fig. 6.8 - 6.9.

6.3.1.3 Depth Map Densification

Next, a 2D depth map is created from all the 3D keypoints in the current keyframe, using a 2D Delaunay triangulation. These 3D kepoints (*i.e.* keypoints observing a known 3D landmark) are used as seeds for a 2D Delaunay triangulation step. The goal of the 2D Delaunay triangulation is to create triangles from the nearest neighbors of each seed by ensuring that no triangle circumcircle contains any other triangle within its surface [1]. This is done by maximizing the smallest angle over the set of all the angles defined by the created triangulation.

[1]Note that, if one takes all these circles' centers and connect them to their three nearest circles centers neighbors we will get the 2D *Voronoi* diagram, which is the dual of 2D Delaunay triangulation.


(A) Initial set of 3D corresponding features between the current and previous keyframes.

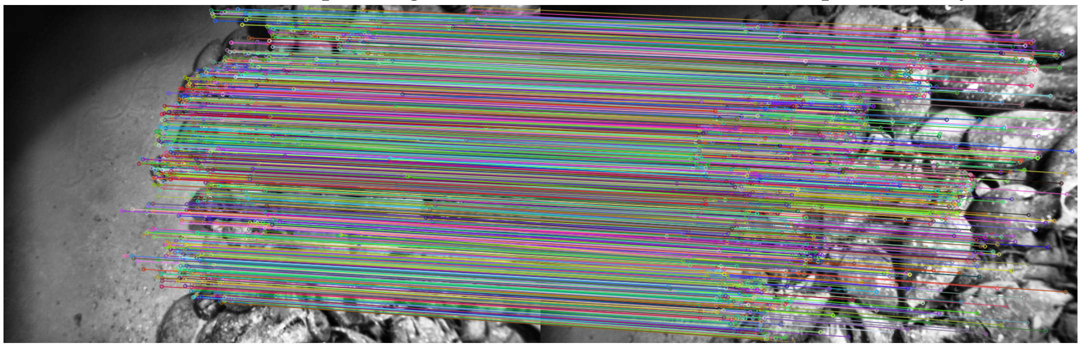
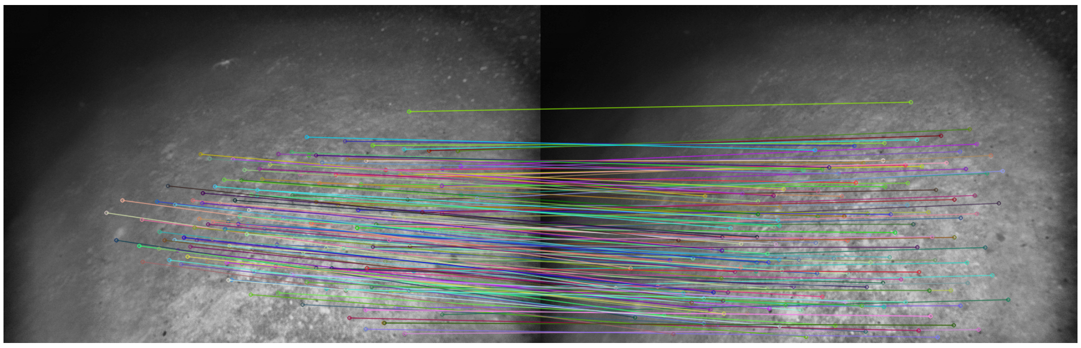

(B) Augmented set of 3D corresponding features between the current and previous keyframes.

FIGURE 6.8: Sample results #1 of 3D augmentation.


(A) Initial set of 3D corresponding features between the current and previous keyframes.

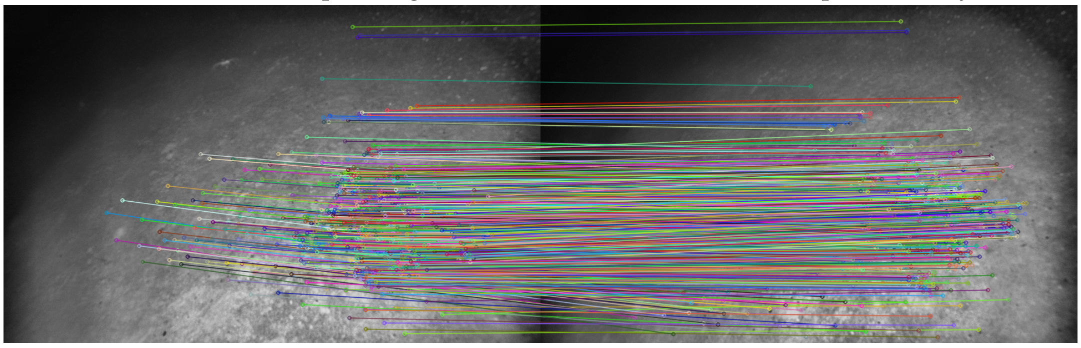

(B) Augmented set of 3D corresponding features between the current and previous keyframes.

FIGURE 6.9: Sample results #2 of 3D augmentation.

The 2D Delaunay triangulation thus returns a set of triangles, whose summits are the provided seeds and whose connected summits correspond to nearest neighbors. While the Delaunay triangulation could be directly performed over the 3D point cloud, we chose to do it in 2D as it highly reduces the complexity of the task.

Once the 2D Delaunay diagram computed, the sparse depth map is densified by interpolating the depth values of each seed over the depth image. This is done by computing the barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$ of every void pixel $\mathbf{p}_i$ within a triangle with summits $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$:

$$\mathbf{p}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T \quad \text{and} \quad \mathbf{s}_j = \begin{bmatrix} u_j & v_j \end{bmatrix}^T \tag{6.8}$$

$$u_i = \lambda_1 \cdot u_1 + \lambda_2 \cdot u_2 + \lambda_3 \cdot u_3 \tag{6.9}$$

$$v_i = \lambda_1 \cdot v_1 + \lambda_2 \cdot v_2 + \lambda_3 \cdot v_3 \tag{6.10}$$

with:

$$\lambda_1 = \frac{(v_2 - v_3)(u_i - u_3) + (u_3 - u_2)(v_i - v_3)}{(v_2 - v_3)(u_1 - u_3) + (u_3 - u_2)(v_1 - v_3)} \tag{6.11}$$

$$\lambda_2 = \frac{(v_3 - v_1)(u_i - u_3) + (u_1 - u_3)(v_i - v_3)}{(v_2 - v_3)(u_1 - u_3) + (u_3 - u_2)(v_1 - v_3)} \tag{6.12}$$

$$\lambda_3 = 1 - \lambda_2 - \lambda_1 \tag{6.13}$$

The depth value d_i of the pixel $\mathbf{p}_i$ is finally computed from the depth values (d_1, d_2, d_3) of $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ as:

$$d_i = \lambda_1 \cdot d_1 + \lambda_2 \cdot d_2 + \lambda_3 \cdot d_3 \tag{6.14}$$

The computation of the dense depth map assumes that neighboring pixels will have similar depth values. In order to limit the effects of this approximation, we add a regularization parameter $\gamma(\cdot)$ to remove pixels which are too far from the triangles' vertices:

$$d_i = \gamma(\mathbf{p}_i) \cdot (\lambda_1 \cdot d_1 + \lambda_2 \cdot d_2 + \lambda_3 \cdot d_3) \tag{6.15}$$

$$\gamma(\mathbf{p}_i) = \begin{cases} 1, & \text{if } \frac{1}{3} \sum_{j=1}^{3} \left\| \mathbf{p}_i - \mathbf{s}_j \right\| \leq \textit{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{6.16}$$

In practice, we have obtained satisfying results when setting the threshold to 20 pixels.

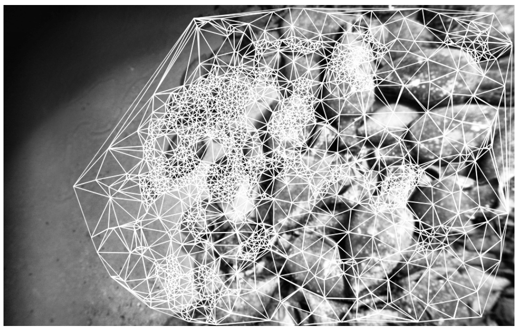
In the context of underwater archaeology, we know that there should not be big depth variations in the 3D scene acquired by the camera. We use this assumption to further regularize the dense 2D depth maps by computing the mean and standard deviation of the depth values within each depth map and remove those farther from the mean than five times the standard deviation. This threshold has been chosen empirically and gives good results on our data. Two examples of densified 2D depth maps are given in Fig. 6.10 - 6.11. These examples illustrate well the impact of the regularization that prevent from taking into account inaccurate areas.

The dense 2D depth map obtained is finally turned into a 3D dense point cloud by projecting every pixel $\mathbf{p}_i$ with a defined depth value d_i into the world frame using the known current keyframe's pose $(\mathbf{R}_{wc}, \mathbf{t}_{wc})$ and camera's calibration matrix $\mathbf{K}$:

$$ {}_w\mathbf{l}_j = \mathbf{R}_{wc} \cdot \mathbf{p}'_i + \mathbf{t}_{wc} \quad \text{with} \quad \mathbf{p}'_i = d_i \cdot \mathbf{K}^{-1} \cdot \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix} \tag{6.17} $$

where ${}_w\mathbf{l}_j \in \mathbb{R}^3$ is the 3D landmark associated to the pixel $\mathbf{p}_i$.

The resulting 3D point clouds are then used in a 3D truncated signed distance field for 3D meshing.


(A) 2D Delaunay triangulation result for Fig 6.8b.

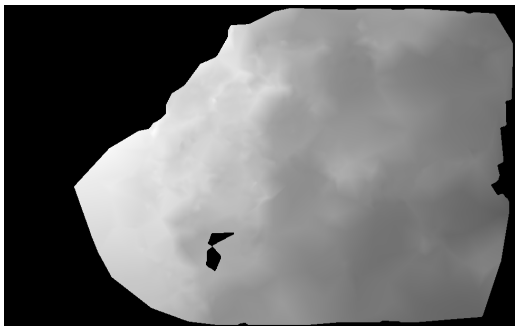
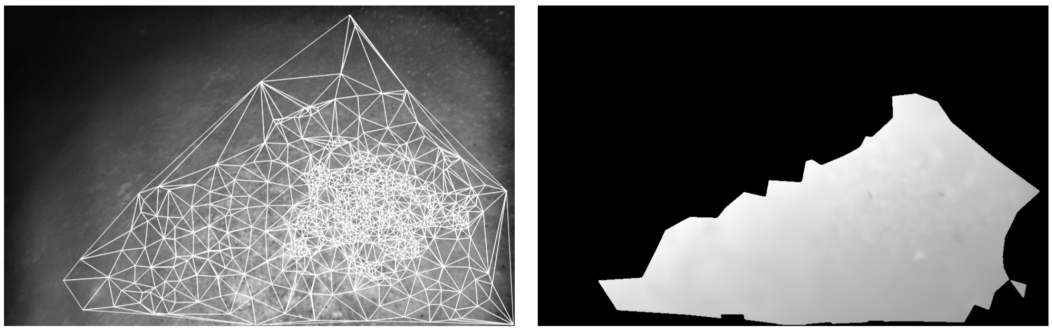

(B) 2D densified depth map from the Delaunay-based interpolation.

FIGURE 6.10: Sample results #1 for depth maps densification.

6.3.2 Dense 3D Meshing

The estimated dense 3D point clouds are then fused within a *Truncated Signed Distance Field* (TSDF). The principle of the TSDF is to discretized the 3D world space into voxels and to assign a signed distance to surface values to each



(A) 2D Delaunay triangulation result for Fig 6.9b.

(B) 2D densified depth map from the Delaunay-based interpolation.

FIGURE 6.11: Sample results #2 for depth maps densification.

voxel (a 2D example of a TSDF is given in Fig. 6.12). The surfaces are defined by the hitting points of the rays projected from the camera, that is by the endpoints of rays defining the projection of the observed 3D landmarks from the camera. The TSDF takes into account the uncertainty of the 3D points to define a hitting probability area and retrieve the associated voxels. For each one of these voxels, a distance value is assigned by computing the distance between the voxel's center and the hitting point of the ray, a positive distance is assigned to voxels between the camera and the 3D point (*i.e.* outside of the surface) and negative distance to voxels behind the 3D point (*i.e.* inside the surface). As we are only interested in close to surface voxels for the 3D reconstruction purpose, the assigned distances which are greater than or lower than a threshold are truncated. This truncation is then used in the reconstruction phase to reduce the computation time by not taking into account voxels whose distance absolute value is equal to this threshold. This threshold is usually set to reflect the expected noise of the provided 3D point clouds.

The voxels' distance values are updated for each new point cloud. The voxels within the hit probability of each ray composing the point cloud are updated by recursively computing an estimate of their signed distance values with a weighted running average. Therefore, for each ray, both the signed distance and a weighting factor of the close voxels are updated. This way, the final signed distance assigned to each voxel takes into account the amount of information received and the noise of the measurements (*i.e.* of the 3D points).

Once the signed distance values are computed, the surfaces of the scene can be extracted by looking for the iso-contours (*i.e.* the 0 values) defined by the voxels. These iso-contours are finally turned into meshes by applying an incremental Marching Cubes algorithm [145].

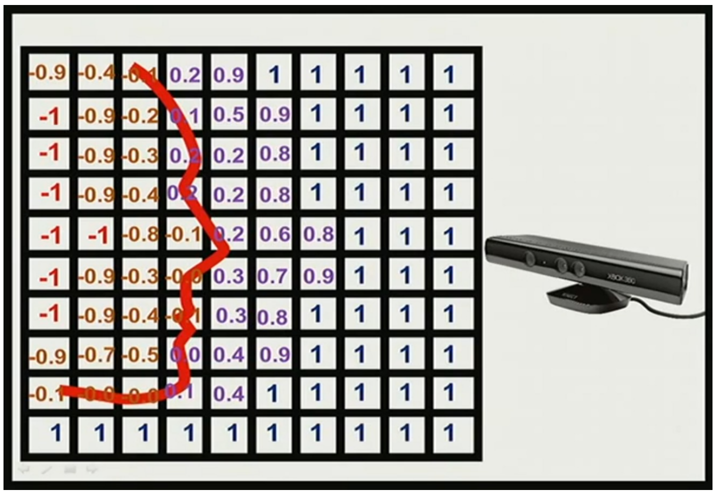


FIGURE 6.12: Example of a 2D TSDF grid (image taken from `http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php`).

In the same way, the TSDF also fuses the pixel intensities related to the 3D points in order to add texture (in our case a color to each vertex) to the 3D meshes.

6.3.3 Implementation Details

We have used the OpenChisel [126] module to implement the TSDF. OpenChisel is a ROS library that implements a TSDF 3D reconstruction. This library is highly optimized to run on CPU in real-time. To do so, it uses a voxel hashing internal representation [171]. This representation dynamically allocates chunk of voxels (*i.e.* small blocks of generally $16 \times 16 \times 16$ voxels) near the estimated surfaces when needed. This way it avoids allocating the whole volume of voxels but stick to the important areas. In order to optimize the voxels' access, a hashing function is used to recover the index of a voxel from its 3D coordinates.

One of the great advantage of such a TSDF 3D reconstruction is that it allows flexible reconstruction which can either focus on accuracy or on speed. For instance, the resolution of the voxels will highly impact the computational load. A maximum distance at which we consider the 3D information to be accurate enough to be processed can also be set in order to avoid reconstructing poorly accurate areas which are far away from the camera. Besides improving the visual representation of the model, this maximum distance also limit the allocation of new chunks of voxels which results in lower memory storage.

As in our case we want the 3D reconstruction to be performed online, we have empirically set the TSDF parameters as follows:

- Chunk size: $16 \times 16 \times 16$ voxels
- Voxel resolution: 0.05 m
- Maximum distance: 4 m
- Expected noise on the 3D data: 0.1 m

In order to test the proposed 3D reconstruction module, we have implemented it in an individual thread that runs in parallel with the SLAM algorithm.

6.3.4 Results

An illustration of the method running online on the archaeological sequence #8 is give in Fig. A.5. The trajectory of the camera is drawn in green and the currently optimized keyframes are shown in blue. Some little black dots appear over the 3D reconstruction and correspond to the 3D landmarks used by the SLAM algorithm. As we can see, the reconstruction is accurate enough to be visually pleasant and exploitable for archaeologists.

In Fig. 6.14 we illustrate the final 3D reconstruction obtained after running the SLAM along with the reconstruction module on the archaeological sequence #9. This illustration highlights well one potential use of this 3D reconstruction for archaeologists: ensuring the full coverage of a wreck. Furthermore, we can see that the accuracy of the 3D reconstruction is sufficient to well distinguish the different amphorae.

On the wreck covered by both sequences, the amphorae are quite massive (at least 30 centimers long) and are hence well above the voxel's resolution set. If one would use the same reconstruction module on a different wreck with smaller objects of interest, the voxel resolution should be set accordingly but at the expense of a heavier computational load. While this should not be a problem on a modern laptop, it could become an issue on the Tegra TX2 embedded in the acquisition systems. As we said, there is a trade-off between real-time performance and accuracy.

6.3.5 Discussion

The proposed dense 3D reconstruction module is pretty simple and yet it works very well in practice. The main limitation is the 3D augmentation step

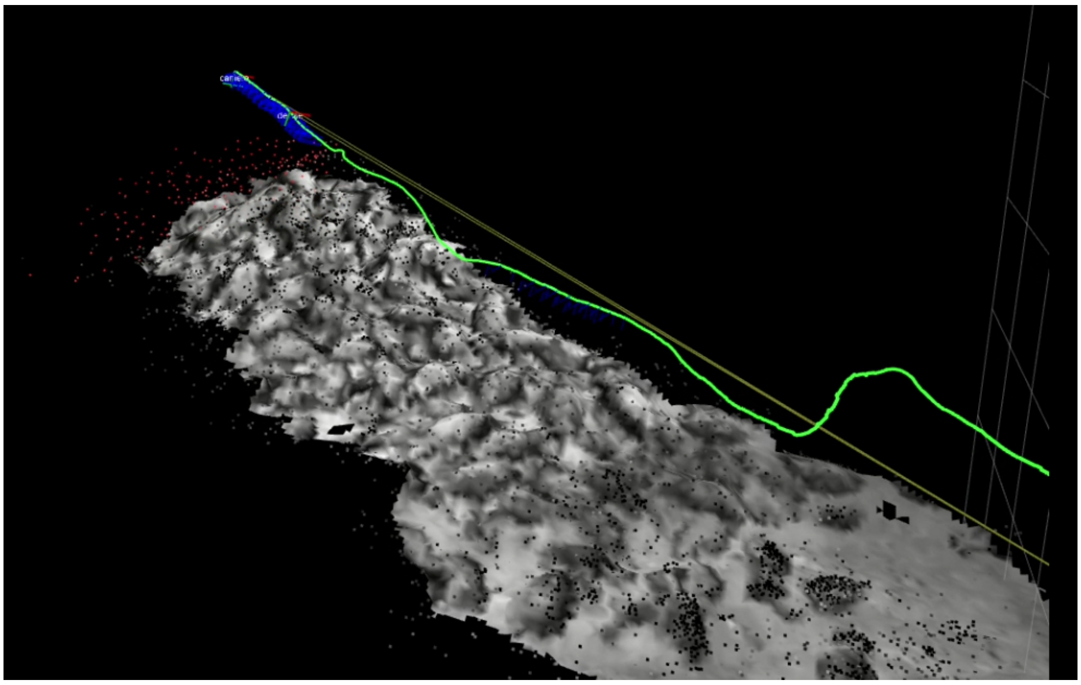


FIGURE 6.13: Online 3D reconstruction for archaeological sequence #8. The trajectory of the camera is drawn in green. The position of the current keyframes in the sliding window are shown in blue.

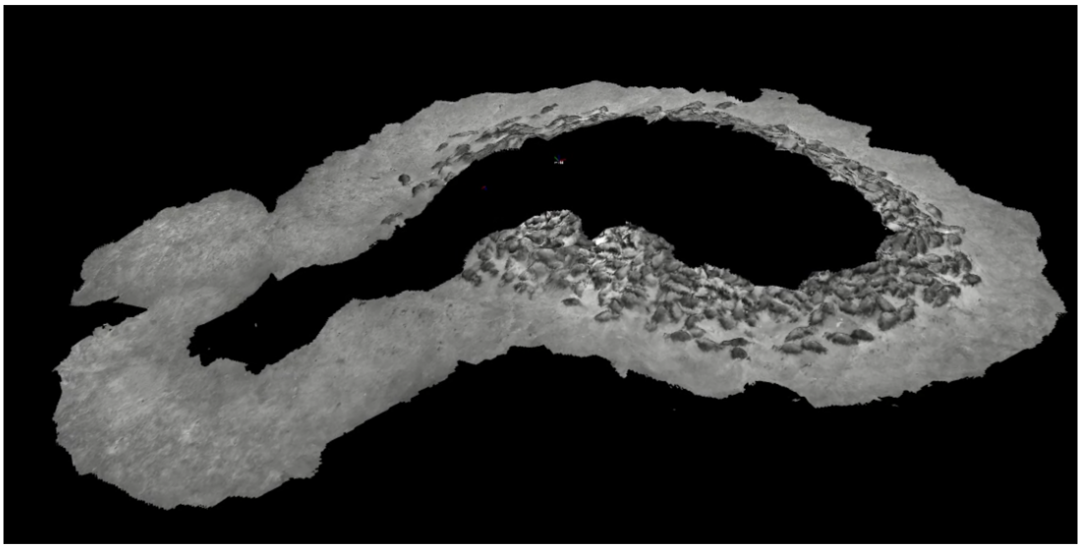


FIGURE 6.14: Final 3D reconstruction for archaeological sequence #9.

which is not optimal. Indeed, there is usually a significant motion of the camera between keyframes and the KLT method is generally not well suited to track features between images that have big pixels displacement. Even if in practice the results are convincing, it would be more adequate to use descriptors at this stage. Descriptors with invariance to rotation and scale would probably be the best candidates as it is the kind of motions that can be expected between keyframes. Also, the regularization steps performed on the 2D depth map are quite empiric and might not be well suited for other environments.

Another limitation is the fact that the 3D reconstruction is purely *static*. This why we have to slightly delay the 3D reconstruction in order to only process 3D landmarks that are fixed or that we consider as accurate enough to be reliable for the purpose of 3D meshing. An enhancement would be to manage moving 3D landmarks in the 3D reconstruction process [181] in order to perform the dense 3D reconstruction at the SLAM's rate.

In the underwater archaeological context, one of the big issue when performing the acquisition of data for offline photogrammetry is to discover later that some parts of the wrecks are missing. The 3D reconstruction module proposed here is therefore very useful to ensure a full coverage of the area of interest. Furthermore, even if the 3D reconstruction quality will depend on the accuracy of the localization, we can imagine performing offline a full Bundle Adjustment at the end of a session in order to optimize the SLAM estimates before running the 3D reconstruction module. In this case, we could also allow more processing to the 3D reconstruction module in order to get finer 3D meshes.

6.4 Conclusion

In this chapter we have presented the design of two acquisition systems that embed a monocular camera, a pressure sensor, a MEMS-IMU and a computer. These acquisitions systems are hence self-contained and their small size make them very easy to embed on almost any ROV. These acquisition systems have been used to record a large dataset that includes a comparative baseline for the purpose of evaluating vision-based SLAM methods. These datasets have been publicly released for the benefit of the community.

Additionally, we have presented a new 3D reconstruction module, compliant with monocular setups. This 3D reconstruction module uses the output of the SLAM algorithm in order to densify its 3D estimations and create a dense 3D meshing from it. The meshing part of this module is done using a truncated signed distance field and is therefore very efficient. This dense 3D reconstruction module runs online but with a slight delay with respect to the SLAM outputs in order to ensure processing only reliable data.

With this new 3D reconstruction module that can be run in parallel of the SLAM algorithm and the design of acquisition systems that can be used to run in real-time all these methods, we have opened the path for online photogrammetry. This application should be of great interest for future underwater archaeology surveys.

Chapter 7

Conclusion

In this thesis we have investigated the problem of localization and mapping in underwater archaeological context. We have proposed a solution based on a vision-based SLAM method that tightly integrates the measurements of a monocular camera, a pressure-sensor and a low-cost MEMS-IMU.

More specifically, the different problems that we have addressed are the following:

How to efficiently process video streams of underwater images for the purpose of localization and mapping ?

We have addressed this problem in chapter 3 by conducting an evaluation of both indirect features tracking methods, based on the use of descriptors, and direct features tracking methods, leveraging on the surrounding pixel intensities, on the specific case of images degraded by typical underwater visual disturbances. We have shown that the Kanade-Lucas-Tomasi (KLT) method, based on the estimation of the optical flow, performed better than other methods for the purpose of features tracking when images are degraded by turbidity or are poorly textured. We have further highlighted that methods based on descriptors have some ambiguity issues that prevent an accurate tracking in such conditions.

We have then used the conclusion of this evaluation to propose a localization and mapping solution which simply requires a monocular camera.

How to accurately ego-localize a robot with only a monocular camera in underwater environments ?

To tackle the problem of monocular localization, we have proposed UW-VO in chapter 4, a keyframe-based Visual SLAM (VSLAM) method that heavily relies on Bundle Adjustment for continuous optimization of both the 3D map and the trajectory. We designed a feature-based monocular VSLAM that uses the KLT method for frame to frame features tracking. We have also proposed a retracking mechanism to overcome a weakness of the KLT. We have shown that this retracking mechanism was helpful during video sequences affected by numerous short occlusions due to moving animals (fishes and shrimps mainly). We have finally evaluated UW-VO along some state-of-the-art monocular VSLAM methods and the results demonstrated that UW-VO was more robust to typical underwater disturbances.

An unavoidable flaw when performing pure monocular VSLAM in an unknown environment is that the scale of the trajectories is not observable. In order to both recover the scale and increase the accuracy of the estimated trajectories, we have then proposed to combine this VSLAM method with a pressure sensor and a low-cost MEMS-IMU.

How to optimally fuse a pressure sensor and a MEMS-IMU with a monocular camera for improved localization ?

In chapter 5, we have first investigated the fusion of the monocular camera with the pressure sensor. We proposed to use a tight fusion paradigm by including the pressure measurements within the nonlinear least-squares optimization of UW-VO (*i.e.* the pose estimation step and the Bundle Adjustment). By delivering depth measurements, the pressure sensor provides a one dimensional absolute information about the vertical position of the system with respect to the water surface. We have shown that this problem was difficult because of a gauge issue due to different frame of references for both sensors (*i.e.* camera and pressure sensor). Different integrations of the depth measurements have been proposed and we have shown that using relative depth error terms within the SLAM method produces better results than absolute depth error terms when there is a significant misalignment between the camera and the depth axis (*i.e.* gravity aligned axis). We have further highlighted that by relaxing the gauge constraint over the orientation of the frame of reference used by the SLAM we could align this frame with the depth axis. We have shown that significant improvement in terms of accuracy could be reached using this method.

The proposed Visual-Pressure SLAM method (UW-VP) produces accurate and scaled trajectories. Yet, it is completely dependent on the visual measurements and fails during short visual information losses. To gain in robustness we further proposed to integrate a MEMS-IMU within the SLAM algorithm.

Following recent advances in the preintegration of IMU measurements, we have proposed to tightly fuse preintegrated IMU measurements within UW-VP. The resulting Visual-Inertial-Pressure SLAM method (UW-VIP) presented similar accuracy in terms of localization compared to UW-VP. However, we have shown that, by integrating inertial measurements, UW-VIP is able to successfully localize during short periods of visual loss, highlighting its higher robustness compared to UW-VP and UW-VO.

The accurate localization and mapping performed by the proposed SLAM methods have then led us to develop a dense 3D reconstruction method, compliant with a monocular camera.

How to get a dense 3D reconstruction from a monocular based setup ?

We have proposed in chapter 6 a dense 3D reconstruction module that leverage the SLAM estimations in order to produce visually accurate 3D meshing. This method uses the keyframes that are not optimized anymore to create dense depth maps by means of interpolation of their 3D observations through a 2D Delaunay triangulation. These depth maps are then integrated into a 3D truncated signed distance field (TSDF) to create the 3D meshes of the reconstruction. The library OpenChisel implements an efficient 3D TSDF method and has been used to render the 3D reconstructions. The fact that the method we propose waits for optimized keyframes create a delay with respect to the SLAM estimations, thus preventing an efficient use for the purpose of trajectory planning or obstacle avoidance. However, we have shown that the 3D reconstructions are accurate enough to be exploited by archaeologists ofr piloting and analysis purpose. While not being as accurate as offline photogrammetry, the proposed method opens the path for online photogrammetry.

All these algorithms have been developed to be embedded on ROVs used in underwater archaeology. Therefore, we have designed two self-contained acquisition systems, that embed a monocular camera, a pressure sensor and a low-cost MEMS-IMU, along with a Tegra TX2 computer, as presented in chapter 6. We have validated the proposed algorithms by successfully running them in real-time on the embedded computer during an archaeological campaign conducted by the DRASSM (Department of underwater archaeology, French Ministry of Culture). We have further taken the opportunity of this campaign to record a large dataset, named AQUALOC, that we have publicly released for the benefit of the community. The data sequences composing AQUALOC contain synchronized measurements suitable to the development of localization and mapping methods. We have also included a comparative baseline trajectory for each sequence by using a Structure-from-Motion library, thus providing a way of quantitatively evaluating the accuracy of different localization methods.

The results of this thesis open the path to very interesting applications. Indeed, the estimation of accurate localization along with the creation of dense 3D reconstruction in real-time provides two information that are highly valuable in other robotic fields such as control and planning, but also to other fields (coral reef monitoring for biologists, for instance). However, the proposed system is still far from being perfect. As vision is the main source of information used in the algorithms, any too strong disturbances for more than a few of seconds would lead to failures. We next discuss about the limitations of the current system and propose some perspectives.

Limitations and Perspectives

A first obvious limitation of the proposed system is a lack of loop-closure capability. Detecting loops in a trajectory is the best way of reducing drift in localization estimations in unknown environments, when no global positioning information is available (such as acoustic positioning systems or GNSS). By detecting a loop at a given point in a trajectory, pose graph optimization can then be performed in order to correct the past trajectory in addition to correcting the current pose. However, as shown in chapter 4, classical techniques based on Bag of Words (BoW) [85] do not work well in underwater environments. The most likely cause of failure is the fact that these BoW are trained offline on big set of images in order to learn what kind of features are good for the problem of relocalization. As underwater images have different characteristics, compared to land or air images, the learnt BoW are very likely to be unsuited. Some works propose to use global descriptors for this task [29, 30]. The use of online BoW [5, 168] also seems to be promising for underwater environments. Indeed, online BoW are not trained online but incrementally built from the processed images. Therefore, the BoW used for the task of loop-closure detection is well adapted to the current environment. While such methods are a bit more demanding in terms of computation, some fast methods, with potential real-time capacity, have been proposed recently [93].

Another limitation, which affects all fusion estimators for the task of localization, is that measurements have to be dropped over time to keep a real-time capability. However this is non-optimal, as the current estimations mainly come from a Maximum Likelihood Estimation over the most recent measurements. A way to deal with this is to rely on marginalization [138]. Marginalizing states instead of dropping them allows to project their covariances on the more recent states, that are kept into the sliding window estimator. The marginalized states then create a prior for the next optimization. However, if not done carefully, this process creates fill-in in the initially sparse Hessian matrix used for optimization, leading to significantly slower optimizations. Yet, by removing measurements that would lead to big fill-in, marginalization can provide a good way for improving the consistency of the estimations. Furthermore, recent methods have shown that Hessian matrix affected by fill-in could be sparsified by setting to zero all the elements that do not provide a significant prior information [109]. The SLAM methods proposed in this thesis could therefore greatly benefit from this kind of enhancement.

Eventually, the use of a monocular camera is kind of a limitation as well. Even if monocular cameras provide enough information for the task of localization and mapping (at least up to a scale factor), they are not as robust as stereo setups [163]. Moreover, when even coupling them with complementary sensors, such as IMUs, the non-observability of scale makes everything more complex. For instance, the estimation of the iMU bias over the accelerometer is a lot easier with stereo cameras thanks to the direct observation of scale [218, 138]. Considering the small size of the acquisition systems designed in this thesis, it would be interesting to embed two of such systems on a ROV in order to create a binocular system [179]. If the sensors measurements could be shared between these two acquisition systems, these latter could be turned into a multi-cameras system with a very large field of view. In practice, such a system with one camera at the front and another at the back of an ROV would be very useful for continuous localization during manipulation tasks. Indeed, durign manipulations, the sediment is moved by the ROV's arm and by the thrusters, which may completely blur the field of view of one of the cameras. But, if installed on the other side of the ROV, the second one could keep a good visibility and continue to provide a localization information. Another use case would be to have one camera set horizontally and one set vertically. Such setups could prove to be useful to localize from both the seabed and potential vertical structures at the same time (in a harbor for instance).

These limitations put apart, the proposed SLAM methods and 3D reconstruction pipeline already provide accurate enough information for many tasks. Besides, SLAM is often a key component in robotic systems for the development of high level features. Given that, the proposed SLAM methods open many perspectives. Multi-temporal navigation (*i.e.* navigation in already mapped areas) [150], multi-robots navigation (with map sharing between the robots) [55, 192] or visual servoing [134, 41] are good examples of scenarios which could benefit from the SLAM outputs.

The rise of deep learning in computer vision has also brought many perspectives for the tasks of localization and mapping in robotics. One could imagine integrating Convolutional Neural Networks (CNNs) within the mapping module to create semantic 3D reconstructions [230, 22, 38], which could both be useful for better a interpretation of the 3D reconstructions, but also as an additional information to integrate within the SLAM for more accurate localization [193, 227, 157]. CNNs have also proven to be efficient for the task of stereo matching [40, 73, 156] or monocular depth estimation [95, 37] and could provide better 3D estimations for localization and mapping. Another obvious application is the use of deep learning for underwater image enhancement (*i.e.* denoising or dehazing) and several works have already be proposed on this topic [141, 140, 67].

Yet, the use of deep learning in underwater scenarios is rather scarce at the time of writing, mainly because of the requirement of large annotated datasets to train CNNs. Yet, we believe that it is just a matter of time before seeing more and more underwater datasets publicly released for this purpose. Moreover, the perspectives offered by unsupervised or weakly supervised learning methods [39, 231] are very interesting in contexts where the acquisition and annotation of large enough datasets is complex.

As a final note, the methods developed within this thesis are actually not tied to the underwater environment and could be used in other fields as well. For instance, the monocular VSLAM method has been successfully tested on video sequences acquired by cameras embedded on UAVs, cars and even airplanes, and its robustness to the challenging underwater visual conditions has proven to be useful in the general case of performing Visual SLAM in visually degraded conditions.

Appendix A

Résumé Étendue

Introduction

Dans cette thèse, nous étudions l'utilisation du SLAM basé vision pour relever le défi de la localisation sous-marine dans un contexte archéologique. Plus spécifiquement, nous étudions d'abord l'efficacité des méthodes de suivi de points d'intérêts dans des images dégradées par des perturbations visuelles typiques en milieux sous-marins. Nous proposons ensuite un algorithme de SLAM visuel (VSLAM) monoculaire, bien adapté aux environnements sous-marins. Cette méthode de SLAM monoculaire est ensuite étendue aux mesures d'un capteur de pression et d'une centrale MEMS-IMU à faible coût. La méthode finale de SLAM Vision-Inertiel-Pression intègre les différentes mesures des capteurs au moyen d'une fusion serrée au sein de l'algorithme VSLAM et fournit des estimations sur l'échelle des trajectoires. Deux systèmes d'acquisition comprenant une caméra monoculaire, un capteur de pression, une MEMS-IMU et un ordinateur embarqué ont été développés pour des expériences sur le terrain. Nous avons également utilisé ces systèmes d'acquisition pour enregistrer un grand jeu de données, qui a été rendu public. En outre, la méthode SLAM développée a été mise en œuvre sur l'ordinateur embarqué et a été testée avec succès en temps réel. Enfin, nous proposons une méthode de reconstruction 3D monoculaire dense en qui fonctionne en ligne.

Cette thèse est organisée comme suit:

1. La première partie détaille l'évaluation des méthodes de suivi de points d'intérêts sur des séquences d'images sous-marines.
2. La deuxième partie présente l'algorithme de SLAM visuel monoculaire que nous proposons.

3. La troisième partie détaille nos travaux sur la fusion d'un capteur de pression et d'une MEMS-IMU au sein de l'algorithme de SLAM visuel.

4. La dernière partie présente les systèmes d'acquisitions que nous avons conçus et le jeu de données que nous avons enregistré avec, ainsi qu'une méthode de reconstruction 3D adapté aux systèmes monoculaires.

1. Évaluation du suivi de points d'intérêts en milieux sous-marins

Le problème du suivi de points d'intérêts est l'une des pierres angulaires de la localisation visuelle. En effet, le SLAM visuel (VSLAM) repose sur le fait que, en observant des points 3D réels dans des images 2D successives (avec un petit mouvement entre eux), il est possible d'estimer la pose (position et orientation) de la caméra par rapport à ces observations 3D. Un point clé pour une localisation précise consiste donc à trouver la projection de ces points 3D dans les images capturées. En outre, si l'on est capable de déterminer les emplacements d'un point 3D inconnu entre deux images prises à partir de deux points de vue différents, ces deux observations peuvent être exploitées pour estimer la position de ce point 3D. Comme l'estimation de la pose de la caméra à partir d'observations 3D connues et la cartographie 3D sont les éléments constitutifs de tout système VSLAM, un suivi très précis des points d'intérêts est requis. Ce problème de suivi de points d'intérêts est un cas particulier de ce qu'on appelle *association de données* dans la communauté robotique.

Cette section est consacrée à l'évaluation de méthodes directes et indirectes pour le suivi de points d'intérêts. Plus précisément, nous comparons la robustesse du suivi de points d'intérêts sur des images sous-marines présentant des effets de turbidité et peu de texture.

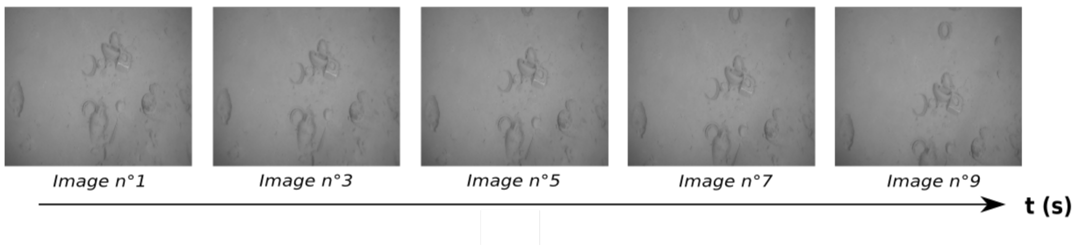


FIGURE A.1: Images prises sur une épave antique (profondeur: 500 mètres, Corse, France) - Credit: DRASSM (Département de Recherche en Archéologie Sub-aquatique et Sous-Marine).

Nous avons donc comparé des méthodes basées sur l'utilisation de descripteurs (méthodes indirectes) à une méthode basée sur le calcul du flot optique (méthode directe) [21]. Les résultats obtenus nous ont montré que la méthode basée flot optique était plus robuste que les méthodes indirectes quand les images présentent des dégradations typiques de l'environemment sous-marin.

2. SLAM Visuel Monoculaire Robuste en Environemments Sous-marin

Cette partie présente UW-VO, la méthode de SLAM visuel monoculaire que nous proposons. Cette méthode est basée sur la sélection d'images clés et utilise fortement des techniques d'ajustement de faisceaux pour optimiser à la fois la carte 3D et la trajectoire estimée.

Au vu des résultats de la section précédente, nous avons appliqué une technique de suivi de point d'intérêts par flot optique au sein d'UW-VO. Les points suivis servent ensuite à construire la carte 3D de l'environemment dans lequel la caméra évolue, et à estimer la pose courante de la caméra. Afin de fonctionner en temps-réel, UW-VO est divisé en deux threads: un thread de suivi de points et d'estimation de pose à la fréquence caméra et un thread gérant les tâches plus complexes et plus lourdes en temps de calcul de cartographie et d'optimization. Le fonctionnement d'UW-VO est résumé dans le schéma suivant :

Nous avons comparé UW-VO à des méthodes de l'état de l'art en SLAM visuel monoculaire, à savoir ORB-SLAM [161], LSD-SLAM [62] et SVO [81], et nous avons montré qu'UW-VO est plus robuste et plus précis que ces méthodes sur des séquences vidéos prises en milieux sous-marins.

Nous avons ensuite étendue UW-VO de façon à fusionner les mesures d'un capteur de pression et d'une IMU-MEMS.

3. SLAM Vision-Pression-Inertiel pour une localisation sous-marine robuste

Une limitation d'UW-VO est que l'échelle métrique des trajectoires n'est pas observable. Cela est dû au fait qu'UW-VO est une méthode qui utilise uniquement une caméra et n'a donc pas d'information directe sur la 3D des scènes imagées. De plus, cette méthode est complètement dépendante des

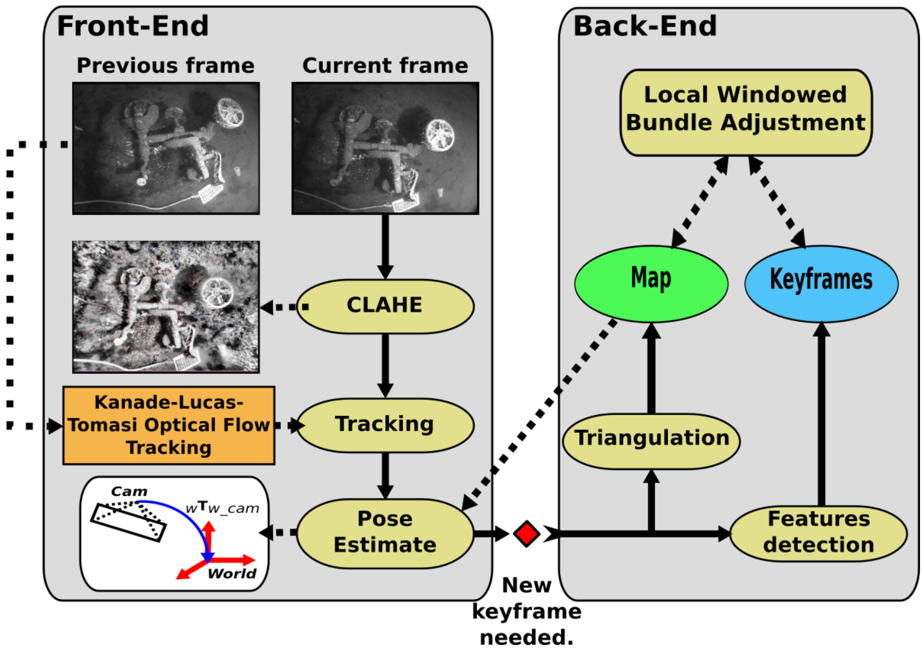


FIGURE A.2: Schéma block représentant le fonctionnement d'UW-VO.

conditions visuelles et est mise en défaut quand ces dernières sont trop mauvaises.

Pour palier cela, nous proposons une fusion serrée des mesures d'un capteur de pression et d'une IMU-MEMS. Le capteur de pression fournit une information sur la profondeur et l'IMU fournit des informations sur l'accélération et la vitesse angulaire du capteur (de façon assez bruité et avec de large biais qui évolue dans le temps).

Nous proposons d'inclure les mesures de ces capteurs dans le graphe de facteurs défini par l'ajustement de faisceaux d'UW-VO. Les mesures du capteur de pression sont inclues en tant que mesures absolues et les mesures de l'IMU sont préintégrées afin de former une seule mesures, facilement intégrable dans le graphe de facteur.

Une fenêtre glissante est utilisée afin d'optimiser le problème d'estimation de pose à partir des différents capteurs. Cette optimisation est réalisée en grâce à la méthode de Levenberg-Marquardt.

Cette méthode rend l'échelle des trajectoires observable et permet d'être robuste à de courtes pertes de visibilité grâce aux informations fournies par le capteur de pression et l'IMU.

Cette méthode de SLAM a pu être testée en conditions réelles grâce à la conception de systèmes d'acquisition.

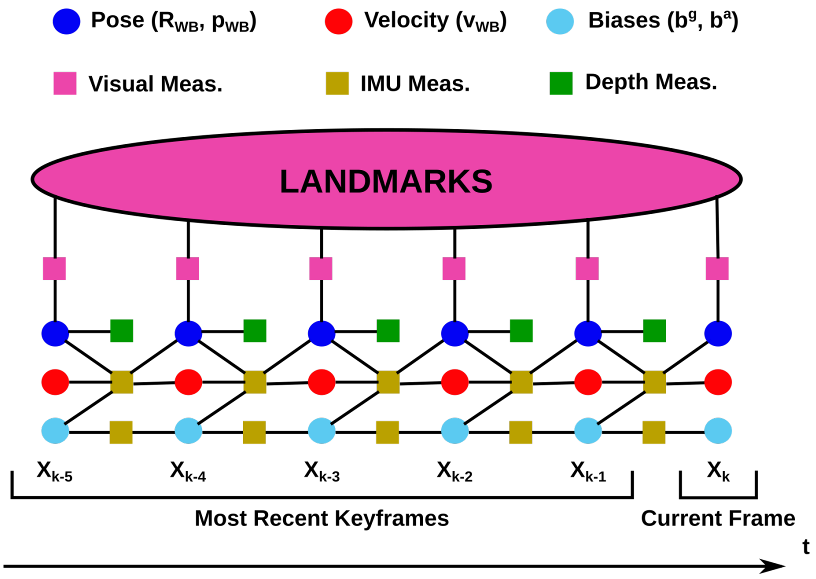


FIGURE A.3: Graphe de facteurs représentant le problème de SLAM Vision-Pression-Inertiel.

4. Systèmes d'acquistion et Reconstruction 3D

Afin de valider l'utilisation des méthodes de SLAM proposées en conditions réelles nous avons conçus deux systèmes d'acquisitions, embarquant les capteurs requis ainsi qu'un ordinateur embarqué. Ces systèmes sont donc autonomes dans le sens où ils peuvent être embarqués sur n'importe quel robot sous-marin. De plus, du fait de leur faible volume, ils sont très facilement embarquables.

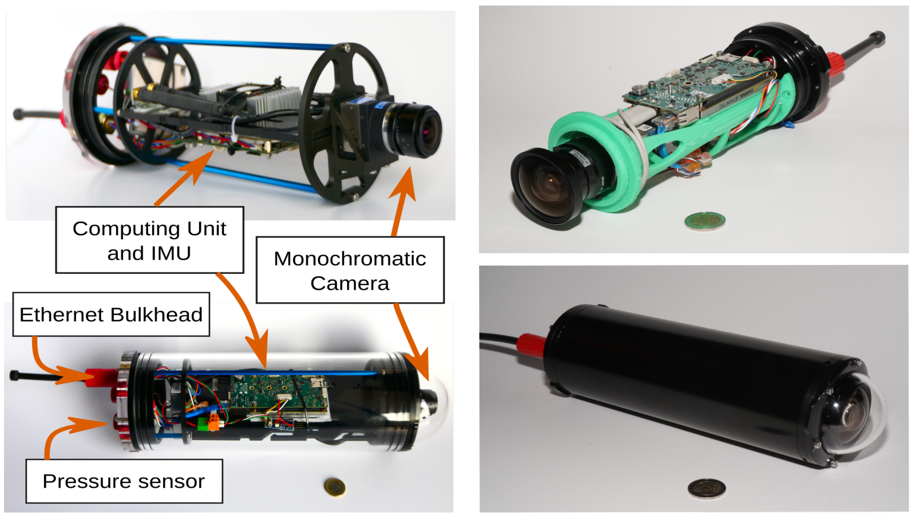


FIGURE A.4: Les systèmes d'acquistion.

En plus d'être utilisé pour valider nos algorithmes, nous avons enregistré un large jeu de données avec ces systèmes. Ce jeu de données a été mis à disposition de la communauté sous le nom d'AQUALOC [74, 72]. Les séquences contenues dans AQUALOC sont adaptés au dévelopement de nouvelles méthodes de SLAM basée vision monoculaire.

Nous avons également exploité les données d'AQUALOC pour développer une méthode de reconstruction 3D dense, adapté aux systèmes monoculaires. Cette méthode produit des cartes de profondeurs dense à partir des estimations 3D de l'algorithme de SLAM. Ces cartes de profondeurs sont ensuite intégrées dans une carte de distance signée 3D afin de créer le meshage 3D.

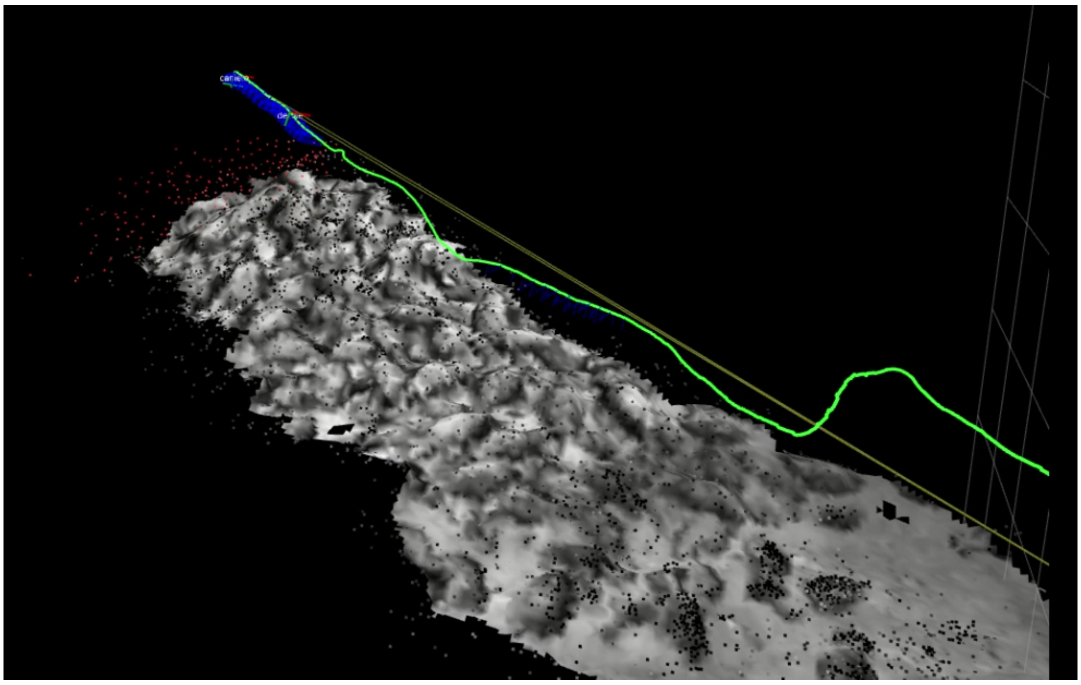


FIGURE A.5: Reconstruction 3D en ligne.

Conclusion

Dans cette thèse, nous avons proposé des solutions au problème de localisation et de cartographie de précision en environemment sous-marin. Plus précisément, nous avons développé une méthode de localisation basée sur un algorithme de SLAM monoculaire, robuste en milieu sous-marin. Cette méthode à ensuite été étendue pour intégrer de façon serré les mesures d'un capteur de pression et d'une IMU à faible coût. Nous avons montré que la méthode de SLAM Vision-Inertiel-Pression qui en a suivi permet d'estimer des trajectoires à l'échelle et d'être robuste à des pertes de visibilité temporaires. Nous avons testé ces algorithmes de SLAM en conditions réelles grâce à la conception de deux systèmes d'acquisitions. Nous avons également utilisé ces systèmes d'acquisition afin d'enregistrer un large jeu de données, que nous avons rendu public. Finalement, à partir des données acquises et des estimations du SLAM, nous avons proposé une méthode de reconstruction 3D dense, adaptée à des systèmes monoculaires et fonctionnant en ligne.

Bibliography

[1] P.-A. Absil, C.G. Baker, and K.A. Gallivan. "Trust-Region Methods on Riemannian Manifolds". In: *Foundations of Computational Mathematics*. Vol. 7. 3. 2007, pp. 303–330. DOI: 10.1007/s10208-005-0179-9.

[2] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. "Bundle Adjustment in the Large". In: *ECCV 2010*. Springer, 2010, pp. 29–42.

[3] A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast Retina Keypoint". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Providence, RI, USA, 2012, pp. 510–517.

[4] Analog. *ADIS16448 IMU*. https://www.analog.com/en/parametricsearch/11172. [Online; accessed 16-Nov-2019]. 2019.

[5] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. "A fast and incremental method for loop-closure detection using bags of visual words". In: *IEEE Transactions on Robotics* (2008), pp. 1027–1037.

[6] Dinesh Atchuthan, Angel Santamaria-Navarro, Nicolas Mansard, Olivier Stasse, and Joan Solà. "Odometry Based on Auto-Calibrating Inertial Measurement Unit Attached to the Feet". In: *2018 European Control Conference (ECC)*. IEEE. 2018, pp. 3031–3037.

[7] J. Aulinas, M. Carreras, X. Llado, J. Salvi, R. Garcia, R. Prados, and Y. R. Petillot. "Feature extraction for underwater visual SLAM". In: *OCEANS 2011 IEEE - Spain*. Santander, Spain, 2011, pp. 160–167.

[8] Josep Aulinas, Yvan R Petillot, Xavier Lladó, Joaquim Salvi, and Rafael Garcia. "Vision-based underwater SLAM for the SPARUS AUV". In: *Proceedings of the 10th International Conference on Computer and IT Applications in the Maritime Industries. Germany*. 2011, pp. 171–179.

[9] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *International Journal of Computer Vision (IJCV)*. Vol. 56. 3. 2004, pp. 221–255.

[10] Timothy D. Barfoot. *State Estimation for Robotics*. 1st. Cambridge University Press, 2017. ISBN: 1107159393, 9781107159396.

[11] Stephen Barkby, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. "A featureless approach to efficient bathymetric SLAM using distributed particle mapping". In: *Journal of Field Robotics* 28.1 (2011), pp. 19–39.

[12] Stephen Barkby, Stefan Williams, Oscar Pizarro, and Michael Jakuba. "An efficient approach to bathymetric SLAM". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 219–224.

[13] Stephen Barkby, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. "Bathymetric particle filter SLAM using trajectory maps". In: *The International Journal of Robotics Research* 31.12 (2012), pp. 1409–1430.

[14] Luca Baroffio, Alessandro E. C. Redondi, Marco Tagliasacchi, and Stefano Tubaro. "A survey on compact features for visual content analysis". In: *APSIPA Transactions on Signal and Information Processing*. Vol. 5. Cambridge University Press, 2016, e13. DOI: 10.1017/ATSIP.2016.13.

[15] H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features". In: *European Conference on Computer Vision (ECCV)*. Graz, Austria, 2006, pp. 404–417.

[16] Fabio Bellavia, Marco Fanfani, and Carlo Colombo. "Selective visual odometry for accurate AUV localization". In: *Autonomous Robots* 41.1 (2017), pp. 133–143.

[17] A. Bender, S. B. Williams, and O. Pizarro. "Autonomous exploration of large-scale benthic environments". In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 390–396.

[18] Jose-Luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. University of Malaga, 2010.

[19] Jose-Luis Blanco-Claraco, Francisco-Angel Moreno-Duenas, and Javier Gonzalez-Jimenez. "The Malaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario". In: *The International Journal of Robotics Research* 33.2 (2014), pp. 207–214.

[20] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. "Robust visual inertial odometry using a direct EKF-based approach". In: *2015 IEEE/RSJ Intelligent Robots and Systems (IROS)*. Hamburg, Germany, 2015, pp. 298–304.

[21] J-Y. Bouguet. "Pyramidal implementation of the Lucas Kanade feature tracker". In: *Intel*. 2000.

[22] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks". In: *Computers & Graphics* 71 (2018), pp. 189–198.

[23] A. Burguera, F. Bonin-Font, and G. Oliver. "Trajectory-Based Visual Localization in Underwater Surveying Missions". In: *Sensors*. Vol. 15. 1. 2015, pp. 1708–1735.

[24] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* (2016). DOI: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html. URL: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

[25] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics (T-RO)*. Vol. 32. 6. 2016, pp. 1309–1332.

[26] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. "BRIEF: Computing a Local Binary Descriptor Very Fast". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*. Vol. 34. 7. 2012, pp. 1281–1298.

[27] Ricard Campos, Rafael Garcia, Pierre Alliez, and Mariette Yvinec. "A surface reconstruction method for in-detail underwater 3D optical mapping". In: *The International Journal of Robotics Research* 34.1 (2015), pp. 64–89.

[28] Ricard Campos, Nuno Gracias, and Pere Ridao. "Underwater multi-vehicle trajectory alignment and mapping using acoustic and optical constraints". In: *Sensors* 16.3 (2016), p. 387.

[29] P. L. N. Carrasco, 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)*. Stockholm, Sweden, 2016, pp. 2589–2595.

[30] Pep Lluis Negre Carrasco, Francisco Bonin-Font, and Gabriel Oliver-Codina. “Global image signature for visual loop-closure detection”. In: *Autonomous Robots* 40.8 (2016), pp. 1403–1417.

[31] Pep Lluis Negre Carrasco, Francisco Bonin-Font, and Gabriel Oliver Codina. “Stereo Graph-SLAM for autonomous underwater vehicles”. In: *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 351–360.

[32] M Carreras, P Ridao, J Batlle, and X Cufi. “AUV navigation in a structured environment using computer vision”. In: *IFAC Proceedings Volumes* 36.4 (2003), pp. 229–234.

[33] Marc Carreras, Pere Ridao, Rafael García, and Tudor Nicosevici. “Vision-based localization of an underwater robot in a structured environment”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 1. IEEE. 2003, pp. 971–976.

[34] Julyan HE Cartwright and Oreste Piro. “The dynamics of Runge–Kutta methods”. In: *International Journal of Bifurcation and Chaos* 2.03 (1992), pp. 427–449.

[35] David Caruso. “Improving Visual-Inertial Navigation Using Stationary Environmental Magnetic Disturbances”. PhD thesis. 2018.

[36] David Caruso, Alexandre Eudes, Martial Sanfourche, David Vissière, and Guy Le Besnerais. “Robust indoor/outdoor navigation through magneto-visual-inertial optimization-based estimation”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4402–4409.

[37] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, and Frédéric Champagnat. “On regression losses for deep depth estimation”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 2915–2919.

[38] Marcela Carvalho, Maxime Ferrera, Alexandre Boulch, Julien Moras, Bertrand Le Saux, and Pauline Trouvé-Peloux. “Technical Report: Co-learning of geometry and semantics for online 3D mapping”. In: *arXiv preprint arXiv:1911.01082* (2019).

[39] Rodrigo Caye Daudt, Bertrand Le Saux, Alexandre Boulch, and Yann Gousseau. “Guided anisotropic diffusion and iterative learning for weakly supervised change detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[40] Jia-Ren Chang and Yong-Sheng Chen. “Pyramid Stereo Matching Network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019).

[41] François Chaumette and Seth Hutchinson. “Visual servo control. I. Basic approaches”. In: *IEEE Robotics & Automation Magazine* 13.4 (2006), pp. 82–90.

[42] Yanqing Chen, Timothy A. Davis, and William W. Hager. “Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate”. In: *ACM Transactions on Mathematical Software*. 2008, pp. 1–14.

[43] Javier Civera, Andrew J Davison, and JM Martinez Montiel. “Inverse depth parametrization for monocular SLAM”. In: *IEEE transactions on robotics* 24.5 (2008), pp. 932–945.

[44] Lee E Clement, Valentin Peretroukhin, Jacob Lambert, and Jonathan Kelly. “The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter”. In: *2015 12th Conference on Computer and Robot Vision*. IEEE. 2015, pp. 23–30.

[45] F. Codevilla, J. D. O. Gaya, N. Duarte Filho, and S. S. C. Costa Botelho. “Achieving Turbidity Robustness on Underwater Images Local Feature Detection”. In: *British Machine Vision Conference (BMVC)*. 2015.

[46] Peter Corke, Carrick Detweiler, Matthew Dunbabin, Michael Hamilton, Daniela Rus, and Iuliu Vasilescu. “Experiments with underwater robot localization and tracking”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 4556–4561.

[47] Vincent Creuze. “Monocular Odometry for Underwater Vehicles with Online Estimation of the Scale Factor”. In: *IFAC 2017 World Congress*. 2017.

[48] Brian Curless and Marc Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. 1996, pp. 303–312. ISBN: 0-89791-746-4.

[49] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2007), pp. 1052–1067.

[50] Frank Dellaert and Michael Kaess. "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[51] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 224–236.

[52] Pierre Drap, Julien Seinturier, Bilal Hijazi, Djamal Merad, Jean-Marc Boi, Bertrand Chemisky, Emmanuelle Seguin, and Luc Long. "The ROV 3D Project: Deep-sea underwater survey using photogrammetry: Applications for underwater archaeology". In: *Journal on Computing and Cultural Heritage (JOCCH)* 8.4 (2015), p. 21.

[53] Pierre Drap, Djamal Merad, Bilal Hijazi, Lamia Gaoua, Mohamad Nawaf, Mauro Saccone, Bertrand Chemisky, Julien Seinturier, Jean-Christophe Sourisseau, Timmy Gambin, et al. "Underwater photogrammetry and object modeling: a case study of Xlendi Wreck in Malta". In: *Sensors* 15.12 (2015), pp. 30351–30384.

[54] A. C. Duarte, G. B. Zaffari, R. T. S. da Rosa, L. M. Longaray, P. Drews, and S. S. C. Botelho. "Towards comparison of underwater SLAM methods: An open dataset collection". In: *MTS/IEEE OCEANS - Monterey*. Monterey, CA, USA, 2016, pp. 1–5.

[55] Rodolphe Dubois, Alexandre Eudes, and Vincent Frémont. "On Data Sharing Strategy for Decentralized Collaborative Visual-Inertial Simultaneous Localization And Mapping". In: *2019 IEEE/RSJ Intelligent Robots and Systems (IROS)*. 2019.

[56] Matthew Dunbabin, Kane Usher, and Peter Corke. "Visual motion estimation for an autonomous underwater reef monitoring robot". In: *Field and Service Robotics*. Springer. 2006, pp. 31–42.

[57] Ethan Eade and Tom Drummond. "Scalable monocular SLAM". In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*. IEEE Computer Society. 2006, pp. 469–476.

[58] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. "Analysis and modeling of inertial sensors using Allan variance". In: *IEEE Transactions on instrumentation and measurement* 57.1 (2007), pp. 140–149.

[59] Armagan Elibol, Nuno Gracias, and Rafael Garcia. "Augmented state–extended Kalman filter combined framework for topology estimation in large-area underwater mapping". In: *Journal of Field Robotics* 27.5 (2010), pp. 656–674.

[60] Armagan Elibol, Jinwhan Kim, Nuno Gracias, and Rafael Garcia. "Fast underwater image mosaicing through submapping". In: *Journal of Intelligent & Robotic Systems* 85.1 (2017), pp. 167–187.

[61] Emcore. *EN-1000-MINAV*. `http://www.cqhongyu6.com/chadhtml/wp-content/uploads/2018/07/EMCORE-OrionTM-EN-1000-MINAV.pdf`. [Online; accessed 16-Nov-2019]. 2019.

[62] J. Engel, T. Schops, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. Zurich, Switzerland, 2014, pp. 834–849.

[63] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[64] Ryan Eustice, Hanumant Singh, John J Leonard, Matthew R Walter, and Robert Ballard. "Visually Navigating the RMS Titanic with SLAM Information Filters." In: *Robotics: Science and Systems*. Vol. 2005. 2005, pp. 57–64.

[65] Ryan M Eustice, Oscar Pizarro, and Hanumant Singh. "Visually augmented navigation for autonomous underwater vehicles". In: *IEEE Journal of Oceanic Engineering* 33.2 (2008), pp. 103–122.

[66] Ryan M Eustice, Hanumant Singh, John J Leonard, and Matthew R Walter. "Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters". In: *The international journal of robotics research* 25.12 (2006), pp. 1223–1242.

[67] Cameron Fabbri, Md Jahidul Islam, and Junaed Sattar. "Enhancing underwater imagery using generative adversarial networks". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7159–7165.

[68] LE Fabrice, Alain Bertholom, Jan Sliwka, and Luc Jaulin. "Interval SLAM for underwater robots; a new experiment". In: *IFAC Proceedings Volumes* 43.14 (2010), pp. 42–47.

[69] Maurice F Fallon, Michael Kaess, Hordur Johannsson, and John J Leonard. "Efficient AUV navigation fusing acoustic ranging and side-scan sonar". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2398–2405.

[70] Renata Ferrari, David McKinnon, Hu He, Ryan Smith, Peter Corke, Manuel González-Rivero, Peter Mumby, and Ben Upcroft. "Quantifying multiscale habitat structural complexity: a cost-effective framework for underwater 3D modelling". In: *Remote Sensing* 8.2 (2016), p. 113.

[71] Fausto Ferreira, Gianmarco Veruggio, Massimo Caccia, and Gabriele Bruzzone. "Real-time optical SLAM-based mosaicking for unmanned underwater vehicles". In: *Intelligent Service Robotics* 5.1 (2012), pp. 55–71.

[72] Maxime Ferrera, Vincent Creuze, Julien Moras, and Pauline Trouvé-Peloux. "AQUALOC: An Underwater Dataset for Visual-Inertial-Pressure Localization." In: *The International Journal of Robotics Research*. 2019.

[73] Maxime Ferrera, Alexandre Boulch, and Julien Moras. "Fast Stereo Disparity Maps Refinement By Fusion of Data-Based And Model-Based Estimations". In: *3DV*. 2019.

[74] Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, Vincent Creuze, and Denis Dégez. "The Aqualoc Dataset: Towards Real-Time Underwater Localization from a Visual-Inertial-Pressure Acquisition System". In: *IROS Workshop - New Horizons for Underwater Intervention Missions: from Current Technologies to Future Applications*. 2018.

[75] Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Real-Time Monocular Visual Odometry for Turbid and Dynamic Underwater Environments". In: *Sensors*. Vol. 19. 3. 2019.

[76] Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Localisation autonome basée vision d'un robot sous-marin et cartographie de précision". In: *ORASIS*. 2017.

[77] Maxime Ferrera, Julien Moras, Pauline Trouvé-Peloux, and Vincent Creuze. "Odométrie Visuelle Monoculaire en Environnement Sous-Marin". In: *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP)*. 2018.

[78] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM*. Vol. 24. 6. New York, NY, USA: ACM, 1981, pp. 381–395.

[79] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[80] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry". In: *IEEE Transactions on Robotics*. Vol. 33. 1. 2017. DOI: 10.1109/TRO.2016.2597321.

[81] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems". In: *IEEE Transactions on Robotics (T-RO)*. Vol. 33. 2. 2017, pp. 249–265.

[82] P. Furgale, J. Rehder, and R. Siegwart. "Unified temporal and spatial calibration for multi-sensor systems". In: *2013 IEEE/RSJ Intelligent Robots and Systems (IROS)*. Tokyo, Japan, 2013, pp. 1280–1286.

[83] P Furgale, T Barfoot, and G Sibley. "Continuous-Time Batch Estimation Using Temporal Basis Functions ". In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. St. Paul, MN, USA, 2012, pp. 2088–2095.

[84] Paul Furgale, Joern Rehder, and Roland Siegwart. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems ". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, 2013, pp. 1280–1286.

[85] D. Galvez-Lopez and J. D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics (T-RO)*. Vol. 28. 5. 2012, pp. 1188–1197.

[86] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. "Complete solution classification for the perspective-three-point problem". In: *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), pp. 930–943.

[87] R. Garcia and N. Gracias. "Detection of interest points in turbid underwater images". In: *OCEANS 2011 IEEE - Spain*. Santander, Spain, 2011, pp. 1–9.

[88] Rafael García, J Puig, Pere Ridao, and Xavier Cufi. "Augmented state Kalman filtering for AUV navigation". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 4. IEEE. 2002, pp. 4010–4015.

[89] Rafael Garcia, Xevi Cufi, and Marc Carreras. "Estimating the motion of an underwater robot from a monocular image sequence". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 3. IEEE. 2001, pp. 1682–1687.

[90] Rafael García, Xevi Cufí, and Lluís Pacheco. "Image mosaicking for estimating the motion of an underwater vehicle". In: *IFAC Proceedings Volumes* 33.21 (2000), pp. 147–152.

[91] Rafael Garcia, J Batlle, Xavier Cufi, and Josep Amat. "Positioning an underwater vehicle through image mosaicking". In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 3. IEEE. 2001, pp. 2779–2784.

[92] Rafael García, Tudor Nicosevici, Pere Ridao, and David Ribas. "Towards a real-time vision-based navigation system for a small-class UUV". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 1. IEEE. 2003, pp. 818–823.

[93] Emilio Garcia-Fidalgo and Alberto Ortiz. "iBoW-LCD: An Appearance-Based Loop-Closure Detection Approach Using Incremental Bags of Binary Words". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3051–3057. DOI: 10.1109/LRA.2018.2849609.

[94] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[95] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *CVPR*. 2017.

[96] Volker Grabe, Heinrich H Bülthoff, and Paolo Robuffo Giordano. "A comparison of scale estimation schemes for a quadrotor UAV based on optical flow and IMU measurements". In: *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2013, pp. 5193–5200.

[97] Nuno Gracias, Pere Ridao, Rafael Garcia, Javier Escartín, Michel L'Hour, Franca Cibecchini, Ricard Campos, Marc Carreras, David Ribas, Narcís Palomeras, et al. "Mapping the Moon: Using a lightweight AUV to survey the site of the 17th century ship 'La Lune'". In: *2013 MTS/IEEE OCEANS-Bergen*. IEEE. 2013, pp. 1–8.

[98] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[99] Richard I Hartley and Peter Sturm. "Triangulation". In: *Computer vision and image understanding* 68.2 (1997), pp. 146–157.

[100] Jan Hartmann, Jan Helge Klüssendorff, and Erik Maehle. "A comparison of feature descriptors for visual SLAM". In: *2013 European Conference on Mobile Robots*. IEEE. 2013, pp. 56–61.

[101] Jon Henderson, Oscar Pizarro, Matthew Johnson-Roberson, and Ian Mahon. "Mapping submerged archaeological sites using stereo-vision photogrammetry". In: *International Journal of Nautical Archaeology* 42.2 (2013), pp. 243–256.

[102] Juan Hernández, Klemen Istenič, Nuno Gracias, Narcis Palomeras, Ricard Campos, Eduard Vidal, Rafael Garcia, and Marc Carreras. "Autonomous underwater navigation and optical mapping in unknown natural environments". In: *Sensors* 16.8 (2016), p. 1174.

[103] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds". In: *Information Fusion*. Vol. 14. 1. 2013, pp. 57 –77.

[104] Marc Hildebrandt, Christopher Gaudig, Leif Christensen, Sankaranarayanan Natarajan, Patrick Paranhos, and Jan Albiez. "Two years of experiments with the AUV Dagon—A versatile vehicle for high precision visual mapping and algorithm evaluation". In: *Proceedings of the 2012 IEEE/OES Autonomous Underwater Vehicles (AUV), Southampton, UK* (2012), pp. 24–27.

[105] Paul W Holland and Roy E Welsch. "Robust regression using iteratively reweighted least-squares". In: *Communications in Statistics-theory and Methods* 6.9 (1977), pp. 813–827.

[106] Jonathan Horgan, Daniel Toal, Pere Ridao, and Rafael Garcia. "Realtime vision based AUV navigation system using a complementary sensor suite". In: *IFAC Proceedings Volumes* 40.17 (2007), pp. 373–378.

[107] Franz S Hover, Ryan M Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J Leonard. "Advanced perception, navigation and planning for autonomous in-water ship hull inspection". In: *The International Journal of Robotics Research* 31.12 (2012), pp. 1445–1464.

[108] Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, and Michael Kaess. "Information sparsification in visual-inertial odometry". In: *Master Thesis*. CMU. 2018.

[109] Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, and Michael Kaess. "Information sparsification in visual-inertial odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1146–1153.

[110] Invensense. *MPU-9250*. `https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf`. [Online; accessed 16-Nov-2019]. 2019.

[111] L. Jaulin. "A nonlinear set membership approach for the localization and map building of underwater robots". In: *IEEE Transactions on Robotics* 25.1 (2009), pp. 88–98.

[112] L Jaulin. "Localization of an underwater robot using interval constraint propagation". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2006, pp. 244–255.

[113] Luc Jaulin. "Robust set-membership state estimation; application to underwater robotics". In: *Automatica* 45.1 (2009), pp. 202–206.

[114] Luc Jaulin, Frédéric Dabe, Alain Bertholom, and Michel Legris. "A set approach to the simultaneous localization and map building-application to underwater robots." In: *ICINCO-RA (2)*. 2007, pp. 65–69.

[115] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. "Imaging sonar-aided navigation for autonomous underwater harbor surveillance". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 4396–4403.

[116] Matthew Johnson-Roberson, Oscar Pizarro, Stefan B Williams, and Ian Mahon. "Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys". In: *Journal of Field Robotics* 27.1 (2010), pp. 21–51.

[117] Matthew Johnson-Roberson, Mitch Bryson, Ariell Friedman, Oscar Pizarro, Giancarlo Troni, Paul Ozog, and Jon C Henderson. "High-resolution underwater robotic vision-based mapping and three-dimensional reconstruction for archaeology". In: *Journal of Field Robotics* 34.4 (2017), pp. 625–643.

[118] Jongdae Jung, Yeongjun Lee, Donghoon Kim, Donghwa Lee, Hyun Myung, and Hyun-Taek Choi. "AUV SLAM using forward/downward looking cameras and artificial landmarks". In: *2017 IEEE Underwater Technology (UT)*. IEEE. 2017, pp. 1–3.

[119] Kenichi Kanatani. *Statistical optimization for geometric computation: theory and practice*. Courier Corporation, 2005.

[120] Tong Ke and Stergios I Roumeliotis. "An efficient algebraic solution to the perspective-three-point problem". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7225–7233.

[121] Oussama Khatib, Xiyang Yeh, Gerald Brantner, Brian Soe, Boyeon Kim, Shameek Ganguly, Hannah Stuart, Shiquan Wang, Mark Cutkosky, Aaron Edsinger, et al. "Ocean one: A robotic avatar for oceanic discovery". In: *IEEE Robotics & Automation Magazine* 23.4 (2016), pp. 20–29.

[122] Ayoung Kim and Ryan Eustice. "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*. IEEE. 2009, pp. 1559–1565.

[123] Ayoung Kim and Ryan M Eustice. "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency". In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 719–733.

[124] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. Nara, Japan, 2007, pp. 225–234.

[125] Georg Klein and David Murray. "Parallel tracking and mapping for small AR workspaces". In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2007, pp. 1–10.

[126] Matthew Klingensmith, Ivan Dryanovski, Siddhartha S. Srinivasa, and Jizhong Xiao. "CHISEL: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially-Hashed Signed Distance Fields". In: *Robotics: Science and Systems (RSS)*. 2015.

[127] L. Kneip, D. Scaramuzza, and R. Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Colorado Springs, CO, USA, 2011, pp. 2969–2976.

[128] L. Kneip and P. Furgale. "OpenGV: A unified and generalized approach to real-time calibrated geometric vision". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, 2014, pp. 1–8.

[129] Laurent Kneip, Hongdong Li, and Yongduek Seo. "Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability". In: *European Conference on Computer Vision*. Springer. 2014, pp. 127–142.

[130] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. "Using Inertial Sensors for Position and Orientation Estimation". In: *Found. Trends Signal Process.* Vol. 11. 1-2. Now Publishers Inc., 2017, pp. 1–153. DOI: `10.1561/2000000094`. URL: `https://doi.org/10.1561/2000000094`.

[131] Szymon Krupínski, Guillaume Allibert, Minh-Duc Hua, and Tarek Hamel. "An inertial-aided homography-based visual servo control approach for (almost) fully actuated autonomous underwater vehicles". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1041–1060.

[132] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011, pp. 3607–3613.

[133] Matheus Laranjeira, Luc Jaulin, and Sébastien Tauvry. "Building underwater mosaics using navigation data and feature extraction". In: *Reliable Computing* 22.1 (2016), pp. 116–137.

[134] Matheus Laranjeira, Claire Dune, and Vincent Hugel. "Catenary-based visual servoing for tethered robots". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 732–738.

[135] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem". In: *International journal of computer vision* 81.2 (2009), p. 155.

[136] S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *International Conference on Computer Vision (ICCV)*. Barcelona, Spain, 2011, pp. 2548–2555.

[137] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization". In: *Robotics: Science and Systems (RSS)*. 2013.

[138] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. "Keyframe-Based Visual-Inertial Odometry using Nonlinear Optimization". In: *The International Journal of Robotics Research (IJRR)*. Vol. 34. 3. 2015, pp. 314–334.

[139] Alberto Quattrini Li, Adem Coskun, Sean M Doherty, Shervin Ghasemlou, Apoorv S Jagtap, Md Modasshir, Sharmin Rahman, A Singh, Marios Xanthidis, Jason M O'Kane, et al. "Experimental comparison of open source vision-based state estimation algorithms". In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 775–786.

[140] Chongyi Li, Jichang Guo, and Chunle Guo. "Emerging from water: Underwater image color correction based on weakly supervised color transfer". In: *IEEE Signal processing letters* 25.3 (2018), pp. 323–327.

[141] Jie Li, Katherine A Skinner, Ryan M Eustice, and Matthew Johnson-Roberson. "WaterGAN: Unsupervised generative network to enable real-time color correction of monocular underwater images". In: *IEEE Robotics and Automation letters* 3.1 (2017), pp. 387–394.

[142] Mingyang Li and Anastasios I Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711.

[143] Mingyang Li and Anastasios I Mourikis. "Improving the accuracy of EKF-based visual-inertial odometry". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 828–835.

[144] Hyon Lim, Jongwoo Lim, and H Jin Kim. "Real-time 6-DOF monocular visual SLAM in a large-scale environment". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 1532–1539.

[145] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. 1987, pp. 163–169. ISBN: 0-89791-227-6.

[146] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision (IJCV)*. Vol. 60. 2. 2004, pp. 91–110.

[147] Quan-Tuan Luong and Olivier D Faugeras. "The fundamental matrix: Theory, algorithms, and stability analysis". In: *International journal of computer vision* 17.1 (1996), pp. 43–75.

[148] T. Lupton and S. Sukkarieh. "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions". In: *IEEE Transactions on Robotics*. Vol. 28. 1. 2012, pp. 61–76. DOI: 10.1109/TRO.2011.2170332.

[149] S Lynen, M Achtelik, S Weiss, M Chli, and R Siegwart. "A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation". In: *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 2013.

[150] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A Hesch, Marc Pollefeys, and Roland Siegwart. "Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization." In: *Robotics: Science and Systems*. Vol. 1. 2015.

[151] Ian Mahon, Stefan B Williams, Oscar Pizarro, and Matthew Johnson-Roberson. "Efficient view-based SLAM using visual loop closures". In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1002–1014.

[152] Ian Mahon, Oscar Pizarro, Matthew Johnson-Roberson, Ariell Friedman, Stefan B Williams, and Jon C Henderson. "Reconstructing pavlopetri: Mapping the world's oldest submerged town using stereo-vision". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2315–2321.

[153] Angelos Mallios, Pere Ridao, David Ribas, Francesco Maurelli, and Yvan Petillot. "EKF-SLAM for AUV navigation under probabilistic sonar scan-matching". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 4404–4411.

[154] Angelos Mallios, Eduard Vidal, Ricard Campos, and Marc Carreras. “Underwater caves sonar data set”. In: *The International Journal of Robotics Research* 36.12 (2017), pp. 1247–1251.

[155] Angelos Mallios, Pere Ridao, David Ribas, and Emili Hernández. “Scan matching SLAM in underwater environments”. In: *Autonomous Robots* 36.3 (2014), pp. 181–198.

[156] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4040–4048.

[157] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. “Fusion++: Volumetric object-level slam”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 32–41.

[158] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. “A Comparison of Affine Region Detectors”. In: *International Journal of Computer Vision (IJCV)*. Vol. 65. 1. 2005, pp. 43–72.

[159] Ondrej Miksik and Krystian Mikolajczyk. “Evaluation of local detectors and descriptors for fast feature matching”. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE. 2012, pp. 2681–2684.

[160] Anastasios I Mourikis and Stergios I Roumeliotis. “A multi-state constraint Kalman filter for vision-aided inertial navigation”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3565–3572.

[161] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics (T-RO)*. Vol. 31. 5. 2015, pp. 1147–1163.

[162] R. Mur-Artal and J. D. Tardós. “Visual-Inertial Monocular SLAM With Map Reuse”. In: *IEEE Robotics and Automation Letters*. Vol. 2. 2. 2017, pp. 796–803. DOI: `10.1109/LRA.2017.2653359`.

[163] Raul Mur-Artal and Juan D Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[164] MM Nawaf, P Drap, JP Royer, D Merad, and M Saccone. "Towards guided underwater survey using light visual odometry". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), p. 527.

[165] Mohamad Nawaf, Djamal Merad, Jean-Philip Royer, Jean-Marc Boï, Mauro Saccone, Mohamed Ben Ellefi, and Pierre Drap. "Fast Visual Odometry for a Low-Cost Underwater Embedded Stereo System". In: *Sensors* 18.7 (2018), p. 2313.

[166] Mohamad Motasem Nawaf, Bilal Hijazi, Djamal Merad, and Pierre Drap. "Guided underwater survey using semi-global visual odometry". In: *COMPIT 15th international conference on computer applications and information technology in the maritime industries, Lecce*. 2016, pp. 287–301.

[167] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2320–2327.

[168] Tudor Nicosevici and Rafael Garcia. "Automatic visual bag-of-words for online robot navigation and mapping". In: *IEEE Transactions on Robotics* 28.4 (2012), pp. 886–898.

[169] Tudor Nicosevici, Nuno Gracias, Shahriar Negahdaripour, and Rafael Garcia. "Efficient three-dimensional scene modeling and mosaicing". In: *Journal of Field Robotics* 26.10 (2009), pp. 759–788.

[170] Tudor Nicosevici, Shahriar Negahdaripour, and Rafael Garcia. "Monocular-based 3-D seafloor reconstruction and ortho-mosaicing by piecewise planar representation". In: *Proceedings of OCEANS 2005 MTS/IEEE*. IEEE. 2005, pp. 1279–1286.

[171] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), p. 169.

[172] D. Nister. "An efficient solution to the five-point relative pose problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*. Vol. 26. 6. 2004, pp. 756–770.

[173] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[174] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. "Large-scale image retrieval with attentive deep local features". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3456–3465.

[175] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3400–3407.

[176] Paul Ozog, Giancarlo Troni, Michael Kaess, Ryan M Eustice, and Matthew 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)*. IEEE. 2015, pp. 1137–1143.

[177] Paul Ozog, Matthew Johnson-Roberson, and Ryan M Eustice. "Mapping underwater ship hulls using a model-assisted bundle adjustment framework". In: *Robotics and Autonomous Systems* 87 (2017), pp. 329–347.

[178] Paul Ozog and Ryan M Eustice. "Toward long-term, automated ship hull inspection with visual SLAM, explicit surface optimization, and generic graph-sparsification". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3832–3839.

[179] Mrinal K Paul, Kejian Wu, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. "A comparative analysis of tightly-coupled monocular, binocular, and stereo vins". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 165–172.

[180] L. Paull, S. Saeedi, M. Seto, and H. Li. "AUV navigation and localization: A review". In: *IEEE Journal of Oceanic Engineering*. Vol. 39. 1. 2014, pp. 131–149.

[181] Enrico Piazza, Andrea Romanoni, and Matteo Matteucci. "Real-time CPU-based large-scale 3D mesh reconstruction". In: *2012 IEEE International Conference on Robotics and Automation (ICRA)* (2018).

[182] Oscar Pizarro, Ryan Michael Eustice, and Hanumant Singh. "Large area 3-D reconstructions from underwater optical surveys". In: *IEEE Journal of Oceanic Engineering* 34.2 (2009), pp. 150–169.

[183] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics*. Vol. 34. 4. 2018, pp. 1004–1020.

[184] M. Quigley, K. C., B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. Kobe,Japan, 2009.

[185] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. "Sonar Visual Inertial SLAM of underwater structures". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.

[186] D Ribas, P Ridao, M Carreras, and X Cufi. "An EKF vision-based navigation of an UUV in a structured environment". In: *IFAC Proceedings Volumes* 36.21 (2003), pp. 287–292.

[187] David Ribas, Pere Ridao, Jose Neira, and Juan D Tardos. "SLAM using an imaging sonar for partially structured underwater environments". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 5040–5045.

[188] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. "Underwater SLAM in a marina environment". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 1455–1460.

[189] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. "Underwater SLAM in man-made structured environments". In: *Journal of Field Robotics* 25.11-12 (2008), pp. 898–921.

[190] Pere Ridao, Marc Carreras, David Ribas, and Rafael Garcia. "Visual inspection of hydroelectric dams using an autonomous underwater vehicle". In: *Journal of Field Robotics* 27.6 (2010), pp. 759–778.

[191] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *International Conference on Computer Vision (ICCV)*. Barcelona, Spain, 2011, pp. 2564–2571.

[192] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. "Multiple-robot simultaneous localization and mapping: A review". In: *Journal of Field Robotics* 33.1 (2016), pp. 3–46.

[193] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. "Slam++: Simultaneous localisation and mapping at the level of objects". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 1352–1359.

[194] Joaquim Salvi, Yvan Petillot, and Elisabet Batlle. “Visual SLAM for 3D large-scale seabed acquisition employing underwater vehicles”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 1011–1016.

[195] Martial Sanfourche, Vincent Vittori, and Guy Le Besnerais. “evo: A realtime embedded stereo odometry for mav applications”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 2107–2114.

[196] Simo Särkkä. *Bayesian filtering and smoothing*. Vol. 3. Cambridge University Press, 2013.

[197] J. L. Schönberger and J-M. Frahm. “Structure-from-Motion Revisited”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA). Las Vegas, NV, USA, 2016, pp. 4104–4113.

[198] Tobias Senst, Jonas Geistert, and Thomas Sikora. “Robust local optical flow: Long-range motions and varying illuminations”. In: *IEEE International Conference on Image Processing*. 2016, pp. 4478–4482.

[199] J. Shi and C. Tomasi. “Good features to track”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, 1994, pp. 593–600.

[200] Florian Shkurti, Ioannis Rekleitis, and Gregory Dudek. “Feature Tracking Evaluation for Pose Estimation in Underwater Environments”. In: *2011 Canadian Conference on Computer and Robot Vision*. St. Johns, NL, Canada, 2011, pp. 160–167.

[201] Florian Shkurti, Ioannis Rekleitis, Milena Scaccia, and Gregory Dudek. “State estimation of an underwater robot using visual and inertial information”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 5054–5060.

[202] Hanumant Singh, Jonathan Howland, and Oscar Pizarro. “Advances in large-area photomosaicking underwater”. In: *IEEE Journal of Oceanic Engineering* 29.3 (2004), pp. 872–886.

[203] Hanumant Singh, Chris Roman, Oscar Pizarro, Ryan Eustice, and Ali Can. “Towards high-resolution imaging from underwater vehicles”. In: *The International journal of robotics research* 26.1 (2007), pp. 55–74.

[204] Jan Sliwka, Luc Jaulin, Martine Ceberio, and Vladik Kreinovich. "Processing interval sensor data in the presence of outliers, with potential applications to localizing underwater robots". In: *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE. 2011, pp. 2330–2337.

[205] Jan Sliwka, Fabrice Le Bars, Olivier Reynet, and Luc Jaulin. "Using interval methods in the context of robust localization of underwater robots". In: *2011 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE. 2011, pp. 1–6.

[206] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. "A micro Lie theory for state estimation in robotics". In: *arXiv preprint arXiv:1812.01537*. 2018.

[207] Shiyu Song, Manmohan Chandraker, and Clark C Guest. "Parallel, real-time monocular visual odometry". In: *2013 ieee international conference on robotics and automation*. IEEE. 2013, pp. 4698–4705.

[208] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. "Double window optimisation for constant time visual SLAM". In: *International Conference on Computer Vision (ICCV)*. Barcelona, Spain, 2011, pp. 2352–2359.

[209] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. "Double window optimisation for constant time visual SLAM". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2352–2359.

[210] Hauke Strasdat, J Montiel, and Andrew J Davison. "Scale drift-aware large scale monocular SLAM". In: *Robotics: Science and Systems VI* 2.3 (2010), p. 7.

[211] Hauke Strasdat, José MM Montiel, and Andrew J Davison. "Visual SLAM: why filter?" In: *Image and Vision Computing* 30.2 (2012), pp. 65–77.

[212] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *2012 IEEE/RSJ Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, 2012, pp. 573–580.

[213] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, Joao Barreto, et al. "Camera models and fundamental concepts used in geometric computer vision". In: *Foundations and Trends® in Computer Graphics and Vision* 6.1–2 (2011), pp. 1–183.

[214] Pedro V Teixeira, Michael Kaess, Franz S Hover, and John J Leonard. "Underwater inspection using sonar-based volumetric submaps". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4288–4295.

[215] Carlo Tomasi and Takeo Kanade. "Detection and Tracking of Point Features". In: *International Journal of Computer Vision*. 1991.

[216] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. "Bundle Adjustment — A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372.

[217] S. Umeyama. "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*. Vol. 13. 1991, pp. 376–380.

[218] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. "Direct visual-inertial odometry with stereo cameras". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1885–1892.

[219] Mark VanMiddlesworth, Michael Kaess, Franz Hover, and John J Leonard. "Mapping 3d underwater environments with smoothed submaps". In: *Field and Service Robotics*. Springer. 2015, pp. 17–30.

[220] Nick Weidner, Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. "Underwater cave mapping using stereo vision". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5709–5715.

[221] Stephan Weiss, Markus W Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments". In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, pp. 957–964.

[222] Stefan Williams and Ian Mahon. "Simultaneous localisation and mapping on the great barrier reef". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 2. IEEE. 2004, pp. 1771–1776.

[223] Stefan B Williams, Oscar Pizarro, Jody M Webster, Robin J Beaman, Ian Mahon, Matthew Johnson-Roberson, and Tom CL Bridge. "Autonomous underwater vehicle–assisted surveying of drowned reefs on the shelf edge of the Great Barrier Reef, Australia". In: *Journal of Field Robotics* 27.5 (2010), pp. 675–697.

[224] Stefan B Williams, Oscar Pizarro, Ian Mahon, and Matthew Johnson-Roberson. "Simultaneous localisation and mapping and dense stereoscopic seafloor reconstruction using an AUV". In: *Experimental robotics*. Springer. 2009, pp. 407–416.

[225] Stephan Wirth, Pep Lluis Negre Carrasco, and Gabriel Oliver Codina. "Visual odometry for autonomous underwater vehicles". In: *2013 MTS/IEEE OCEANS-Bergen*. IEEE. 2013, pp. 1–6.

[226] Kejian J Wu, Chao X Guo, Georgios Georgiou, and Stergios I Roumeliotis. "Vins on wheels". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5155–5162.

[227] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. "Mid-fusion: Octree-based object-level multi-instance dynamic SLAM". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5231–5237.

[228] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22 (2000).

[229] Zhengyou Zhang. "Camera calibration". In: *Computer vision: a reference guide* (2014), pp. 76–77.

[230] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. "SceneCode: Monocular Dense Semantic Reconstruction using Learned Encoded Scene Representations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11776–11785.

[231] Chao Zhou, Hong Zhang, Xiaoyong Shen, and Jiaya Jia. "Unsupervised learning of stereo matching". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1567–1575.

[232] K. Zuiderveld. “Contrast Limited Adaptive Histogram Equalization”. In: *Graphics Gems IV*. 1994.