POLITECNICO DI TORINO

Master's Degree in ICT For Smart Societies

Master's Degree Thesis

Unmanned Underwater navigation based on visual and inertial sensors

Supervisors

Prof. Marco PIRAS

Prof. Paolo DABOVE

Dott. Vincenzo DI PIETRA

Candidate

Riccardo RUGGIU

April 2021

Abstract

Unmanned Underwater Vehicles (UUVs) are widely used for decades mainly in exploration missions and underwater maintenance. Autonomous Underwater Vehicles (AUVs) represent a class of this vehicles that is becoming more and more popular but they have high costs due to the price of the sensors needed for autonomous navigation purpose. This thesis has been focused on the estimation of UUV trajectories, combining Visual Odometry (VO) and Inertial Measurement Unit (IMU) approach. The challenge was to apply in an underwater environment methods commonly used in submerged scenarios. Another important aspect of this study was the utilization of low-cost hardware for data acquisition and manipulation. All algorithms were designed to run in a Raspberry Pi, a single camera for VO and an entry level IMU were used. A characterization of IMU was done in order to prevent from gyroscope's drifts over the time. The fusion of VO and IMU measurements was done using an Extended Kalman Filter (EKF). Testing phase was divided in three steps: first tests were centered on performance of feature based VO algorithms in an underwater dataset, improving performance of these using image enhancement techniques. The second phase was focused on VO algorithm with the low-cost hardware, working in a submerged environment. The last phase consisted in a series of underwater tests of the complete algorithm.

Acknowledgements

First thanks goes to all my family, especially to my mother, my father and my sister Chiara, for their constant support through all the past years. Paolo needs a special mention, since he has been more than a cousin and more than a roommate from the beginning of this adventure.
Another big thanks goes to my Turin's friends Dario, Damiano, Federico, Flavio, Giorgio, Giovanni, Giulia, Hilde, Lorenzo, Mara, Massimo, Pompilio, Sebastiano and Valentina whom always delighted the study hours.
Also my distant friends had a key role in this path, especially Antonio, Ary, Bussu, Chiscu, Fede, Michele, Mureddu and Vale, supporting me with long text messages and hours of calls.
All the people inside the Team PoliTOcean need a big thanks, for all the days and nights spent building ROVs and having fun in the laboratory together. Special mention goes to the founder, Dario, who inspired me with his hard work and passion, and to Massimo, for all the "magic tricks" that he has done to make run the prototypes. Angelo, Federico and Natalia need a mention since they represent the "new guard" of the team and I'm sure that they will guide it to new milestones.
Finally a big thanks goes to the supervisors of this thesis for their support and patience.

Table of Contents

List of Tables

List of Figures

Acronyms

VO
Visual Odometry

IMU
Inertial Measurement Unit

INS
Inertial Navigation System

UUV
Unmanned Underwater Vehicle

ROV
Remotely Operated Vehicle

AUV
Autonomous Underwater Veichles

EKF
Extended Kalman Filter

ORB
Oriented FAST and Rotated BRIEF

SIFT
Scale-Invariant Feature Transform

SURF
Speeded Up Robust Features

Chapter 1

Introduction

Underwater Unmanned Vehicles (U.U.V.s) are becoming nowadays more and more popular, since the developments in technologies and the growing interest in underwater exploration. Underwater aided navigation represents still a big issue, since the impossibility to use radio-based signals that are commonly used in submerged scenarios, such as GPS.
To overcome this problem, typically these systems adopts acoustic based sensors which are used to help the navigation process. Main problems of these solutions are the drifts and the noises of the observations.
Thus is preferred to integrate various readings from different sensors/cameras in order to have a more robust system. This implementation is costly since this kind of sensors is quite expensive and not straightforward to integrate.
The objective of this master's thesis was to find an alternative to improve underwater navigation, using low cost hardware. The study was focused on the development of a system able to reconstruct UUV's trajectory during a navigation, combining inputs of the camera and reading from an inertial sensor.
Visual Odometry(VO) was selected for retrieve the path using the camera, paired up with an Inertial Measurement Unit (IMU), with the deployment of an Extended Kalman Filter.
VO based navigation techniques have almost progressed in order to reach a positioning level of accuracy of about few tens of decimeter in terrestrial scenarios, while in underwater environments is still challenging, since the difficulty of have a clear view of the field.
The hardware employed consisted in a *Raspberry Pi 4*, which is a common and affordable embedded computer, a low-cost camera and an entry level IMU. Challenging part of this study was to run the algorithms with this kind of hardware. In order to run in almost real time the localization algorithm, studies were made focusing on the computational time needed by each block. Another important aspect was the characterization of the sensors, needed to figure out if the IMU was

reliable for this kind of application compared to the high-cost ones.
The coding part was written using Python as language and the libraries employed in this project are all open source.
First part of the following work covers the literature review of the problem, focusing on UUVs and VO problem.
In Chapter 4 the low cost sensor's characterization is described in details. Particular attention in this chapter is given to the acquisition part of the IMU's data. In order to improve its performance a library for retrieve IMU's readings was developed, exploiting all its characteristics. Other tests were done on the inertial sensor to measure its drifts over the time.
The technical challenge of the implementation is illustrated in Chapter 5: in this section is presented the development step by step of the algorithm, complemented by three main tests.
First tests were centered on the performance of feature based VO algorithms in an pre-existing underwater dataset, improving performance of these using image enhancement techniques, focusing also on computation time.
Second phase was focused on VO algorithm with the low-cost hardware, working in a submerged environment. From this environment a dataset was created, acquiring images with an ROV and tracking its movement with a GPS antenna. In this phase was integrated also a first version of the EKF, in order to integrate the sensor's fusion, improving VO's performances.
The last phase consisted in an underwater test of the complete algorithm. For this test was developed an apposite waterproof platform (which simulates the ROV), integrating the low-cost hardware selected for this study. In this section is described the creation of the platform and the acquisition done in the underwater environment with the tracking of the truth track of platform's movement. Furthermore is presented the integration of VO and IMU which was done using the EKF, with the results obtained.

Chapter 2

Unmanned underwater vehicles

Oceans cover the 71% of the Earth's surface. They dictate weather conditions, temperature and humidity changes. They provide an habitat for many species and provide energy that can be used by humans. Only nearly 8% of Earth's oceans have been explored. This is caused by the lack of technology for use in exploration.[1]
Unmanned underwater vehicles (UUVs) are any vehicle capable to operate underwater without the human presence inside. UUVs have been used over the past half century, trough military and oil and gas research.
Ocean exploration started in the 60s with the US Navy developing their first UUVs (CURV and CURV II). These systems were developed for rescue and recovery of ordnance.
From the 80s oil and gas industry has driven for research and development of this technology. At the end of 80s UUVs have been used to locate many historic shipwrecks, such as the *Titanic* and the *SS Central America*, recovering from them material. Recently, commercial companies started developing affordable UUVs reducing technology cost. Today, they are manly used in offshore platforms (maintenance tasks on submersed infrastructures), in military field (detonate underwater mines) and in scientific field (exploration and ocean floor mapping).

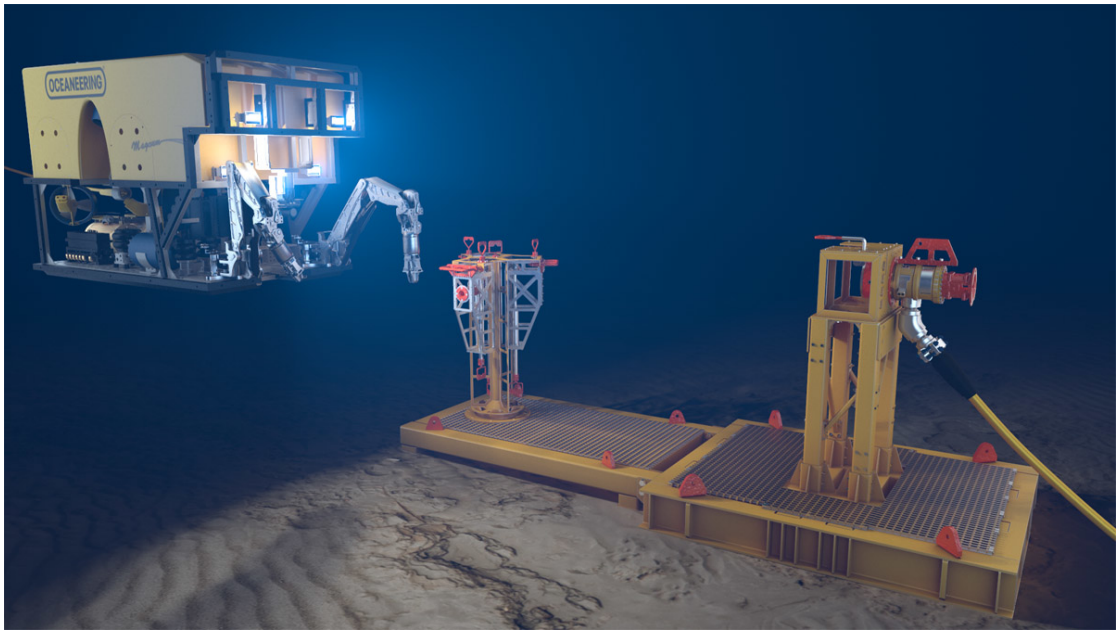

Figure 2.1: UUV at work, credits [2]

UUVs are divided in two main classes: ROVs and AUVs.

2.1 ROVs

Remotely Operated Vehicles (ROVs) are a class of unmanned underwater vehicles. The main characteristic is that the vehicle is attached to the surface through a tether for data communication and, in some cases, also for power. They are equipped with a camera and other sensor useful for the their purpose.
There are four categories of ROVs, based on size/weight and tasks.

- **Class I** called *observation* R.O.V.s. Used for inspection task, they are the smallest ones. Equipped with only a camera and lights, they can reach depths of 300 meters. Usually they are powered by an on-board battery. Their typical weight is up to 40 Kg.

- **Class II** similar to the class I but with the payload option. They can be fitted with several sensors and also with manipulators, such as a robotic arm, in order to perform light maintenance tasks. This class weights from 40 to 300 Kg.

- **Class III** called *work class*. Characterized by their elevate capability of carrying large payloads,for that are used for heavy interventions in high

depths. They can withstand until 10000 meters of depth and they are bigger than class I and class II R.O.V.s. Usually these vehicles are powered from the surface due to the elevate usage of energy. They typically weights more than 300 Kg.

- **Class IV** called *Excavators*. These vehicles operate on the sea beads to do excavations tasks, such as burying of pipelines/cables. They have a limited manoeuvrability, weighing up to 5000 kg.

Figure 2.2: Class I ROV, credits [3]

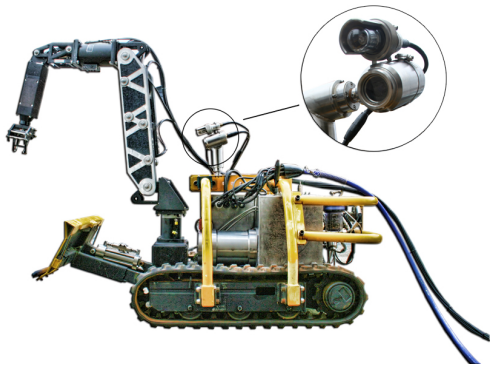

Figure 2.3: Class IV ROV, credits [4]

2.2 AUVs

Autonomous Underwater Veichles (AUVs) are used for underwater survey missions such as detecting and mapping submerged wrecks, rocks, and obstructions that can be a hazard to navigation for commercial and recreational vessels. At present, AUVs are mostly employed in survey applications with larger areas of sea. An A.U.V. conducts its survey mission without operator intervention. When a mission is complete, the AUV will return to a pre-programmed location where the data can be downloaded and processed. It is powered by on-board batteries and it does not have any cable connected to the surface. Navigation is one of the key AUV technologies because the localization, path tracking and control of the vehicle are all based on precise navigation parameters. Some navigation methods commonly used for land and air are not suitable for underwater because of the attenuation effect of water on electromagnetic signals.

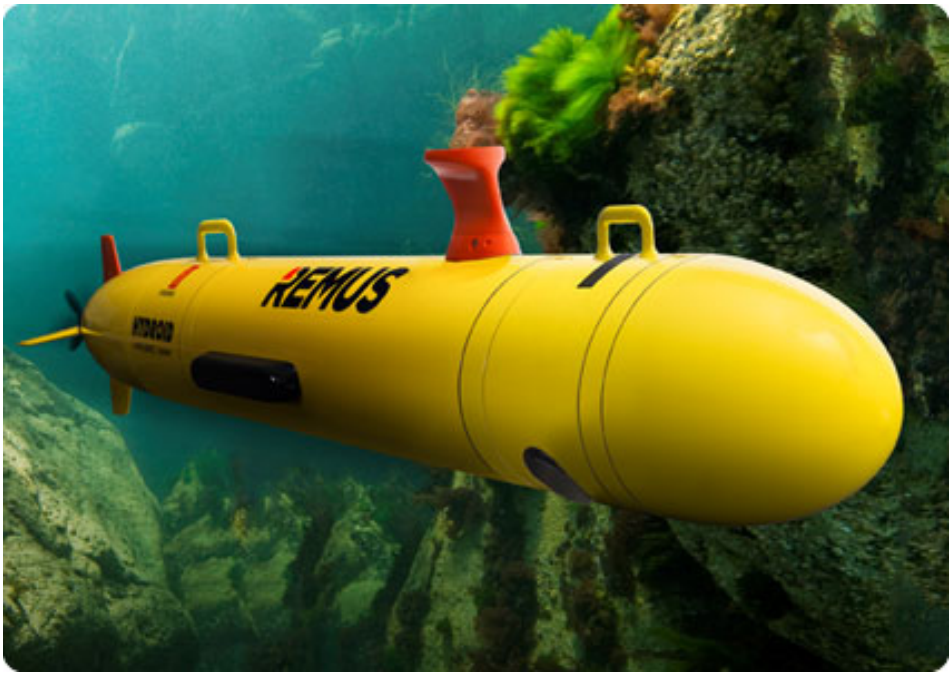

Figure 2.4: AUV, credits [5]

2.3 Sensors overview

UUVs are equipped with various sensors needed for improve navigation tasks. These sensor are specially useful in AUVs, since they need to do navigation task autonomously.
Most common are:

- **IMU** Inertial Measurement Unit (IMU) is a type of sensor very common on UUVs. IMUs are composed of accelerometers, gyroscopes, and, in some cases, magnetometers. Accelerometers are used to measure linear acceleration, gyroscopes to measure angular velocity, magnetometers to measure magnetic field strength in order to estabilish cardinal direction (directional heading). IMU in AUVs is used as central navigation system because of its autonomy. Such sensors can be quite sensitive to noise and disturbances from magnetic fields, resulting in drift. The errors of the IMU increase with increasing elapsed time due to the drift of accelerometers and gyroscopes [6].

- **DVL** Doppler Velocity Log (DVL) is an acoustic sensor that estimates velocity relative to the sea bottom. This is achieved by sending a long pulse along a minimum of three acoustic beams, each pointing in a different direction. Typically, this produces estimates of velocity converted into an XYZ coordinate frame of reference – the DVL's frame of reference. Together with a heading estimate, these velocity estimates may be integrated over the ping interval to estimate a step-by-step change of position [7].

- **Depth Sensor** Subsea level measurement can be accomplished with the use of a pressure sensor. As the UUV is submerged, the water pressure is exerted on the diaphgram of the pressure sensor. The deeper the system is submerged, the higher the pressure. A correlation between pressure and output signal is made to measure the depth of the equipment.

- **Sonar** SOund Navigation And Ranging is a technique that exploits sound propagation underwater in order to detect objects, retrieve position or even to send data (an example of sonar-based modem can be found in [8]). In UUVs, the sonar sensor emits pulsed and listen to the echoes, this is called active sonar technique. Knowing the speed of sound in water, the sensor can calculate the distance the sound has travelled. Scanning sonar are used in UUVs to retrieve distance from surrounding objects and help navigation in low visibility condition, since targets with material densities very different from water (such as gas, rock, concrete or metal) will be very reflective and have strong echoes. Active sonar technique is used also for tracking U.U.V.s. These systems are divided in three classes: Short Baseline (SBL), Ultra-Short Baseline (USBL), and Acoustic Long Baseline (LBL). Transponder beacons are

used either on sea floor or at the surface; the UUV sends a signal that is then returned by the transponder and an estimation of the position is computed based on the round-trip delay. Details of these classes can be found in [9].

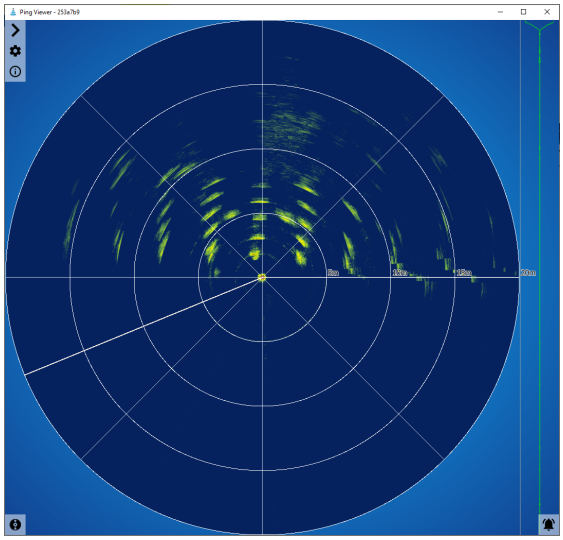

Figure 2.5: UUVs surroundings from scanning sonar, credits [10]

Chapter 3

Visual odometry

3.1 Alternative Positioning Systems

A positioning system is a mechanism for determining the position of an object in space. In outdoor and in-air environments Global Navigation Satellite Systems (GNSS) is the most common and most efficient technology, it can provide an elevated accuracy on position estimation. This system is commonly know as Global Positioning System (GPS), even if GPS is only one of many GNSS systems available. GNSS is based on radio signals, that travel back and forth from the receiver to the satellite. More details about GNSS systems can be found in [11]. Other systems are based on radio beacons or optical signals, such as laser based one. Light Detection and Ranging (LIDAR) works with one laser source and a rotating mirror (called *beam*). It is used to compute the distance to many points around it and generates information about the environment in 2D. To retrieve 3D dense information multi-beam LIDARs are used [12].
In underwater context the previous techniques can not be used, since most of these signals are jammed from the water. Alternative positioning systems for this environment are based on active acoustic, sing imaging scanning Sonar or Doppler Velocity Logs (DVL) previously described. These solutions are expensive since Sonar and DVL sensors require high technical skills for their deployment and operation. Moreover, their size specifications prevent their integration within small mobile systems, and require a high computational power.
Another positioning system is the Intertial Navigation System (INS): it uses the data sampled from the IMU and calculates the position, velocity and heading without need for external references, using a dead reckoning method. More details of INS can be found in [13]. INS requires, to work properly stand alone a reliable IMU, that has small drifts. This category of sensors has an elevated cost. Common IMUs, have large drifts but are very affordable since their reduced cost.

The objective of this study was to find an alternative underwater positioning solution, using low cost hardware. For this reason was selected Visual Odometry fused with INS navigation using a camera for VO and a low cost IMU for INS. The sensors fusion was done with a Kalman Filter.
This chapter describes the theoretical background of this study, focusing of Visual Odometry and Kalman Filter.

3.2 Visual Odometry

Visual Odometry (VO) is defined as the pose estimation of an object that computes a stream of images acquired from a single or multiple cameras. VO provides an incremental online estimation of a veichle's position. Specifically, integrates pixel displacements between image frames over time. VO methods can be divided into two main categories: feature-based methods (based on feature detection over the image) and direct methods, which use all the pixels in the image to detect motion. From results on [14] it is clear that feature-based methods are more robust in underwater environments, therefore in this study feature-based methods were used. Another distinction can be made over the number of cameras used: monocular VO or stereo VO.

- **Stereo VO**: In stereo VO, 3D information about the environment is computed for each time step using 3D triangulation of features in both the right and the left image. The motion is estimated by observing the features in two consecutive time frames in both the right and the left image. Stereo VO methods only require two time-successive frames. Objects scale can also be estimated since the baseline distance between the cameras is known.

- **Monocular VO**: Monocular VO methods only rely on one camera for VO. 3D information extraction requires two time successive frames. Motion estimation requires an additional third frame for transformation calculation. Monocular vision poses extra challenges compared to stereo vision since scale information is not known, it is usually set to a pre-defined value. In this study, **monocular VO was used.**

3.3 Feature-based methods

A feature-based VO method follows these steps: feature detection, feature description, feature matching and motion estimation. The steps can be seen in figure 3.1.

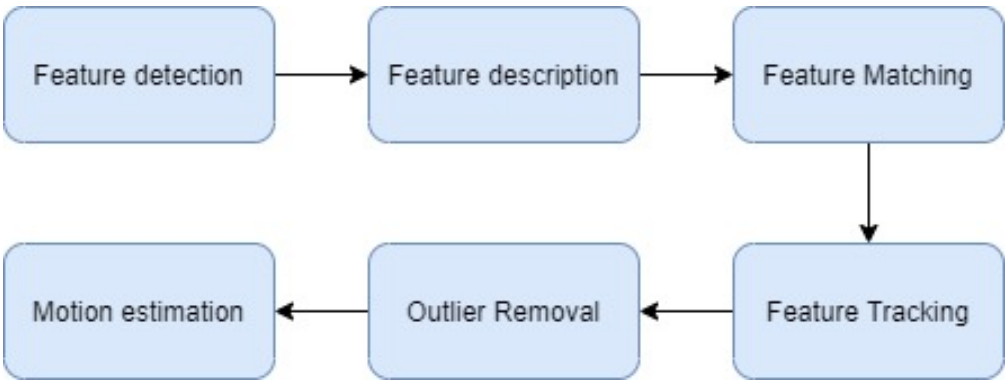

Figure 3.1: Feature-based Visual Odometry steps

There are two methods for represent the image: through global feature or local features. With global features the image is represented by one multidimensional feature vector, with values that measure various global aspect of the image such as colour, texture and shape. This representation is not invariant to significant transformation and is not suited for this study's purpose, so local feature representation was used.
Local feature is defined as a specific pattern which is unique from its immediately close pixels, generally associated with one or more of image properties. Such properties include edges, corners, regions that remain invariant to viewpoint and illumination changes. These local features are then converted into numerical descriptors, representing unique and compact summarization of these local features. Ideal local features properties are listed in [15], most important are robustness and invariancy to transformations such as rotation, scale etc. A trade-off between the properties must be done, in the design of the feature detector. Best possible features in a image are [15]:

- **Edges**: pixel patterns at which the intensities abruptly change, with a strong gradient magnitude.
- **Corners**: points at witch two (or more) edges intersect in the local neighbor.
- **Regions**: a closed set of connected points with a similar homogeneity criteria (such as the intensity value).

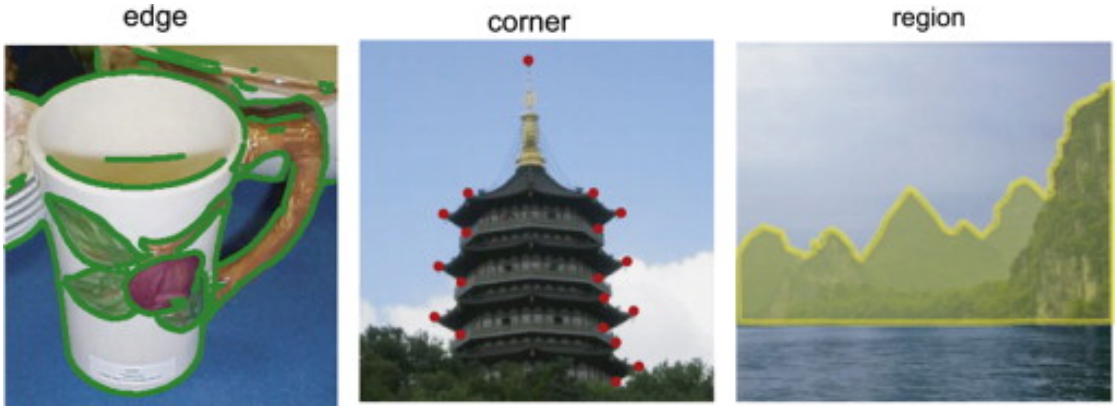

Figure 3.2: Example of local features, credits [16]

3.3.1 Feature Detectors

A feature detector is an algorithm that spots in the image the local features, that are likely to be detected again in a similar image. Feature detectors (and extractors) must be designed finding a trade off among the set of their properties, based on the use case. Main properties of feature detectors are listed in [17]. Existing feature detectors can be divided in three main categories: single scale detectors, multi scale detectors and affine invariant detectors.

Single Scale Detectors

Single scale detectors are invariant to image's transformations such as rotation, translation, change in illumination, addition of noise, but not invariant to scale. Most important single scale detectors are:

- **Harris detector**: based on Moravel's corner detector (described in [17]), combines corner and edge detection by obtaining the variation of auto-correlation between adjacent regions over all different orientations. It is invariant to rotation and illumination changes.
- **Shi Tomasi detector**: also called GftT (Good Features To Track) is a corner detector based on Harris detector with a modification in the scoring function: it uses the minimum eigenvalue of each $2x2$ gradient matrix to detect "good features", as described in [18].
- **FAST detector**: Features From Accelerated Segment Test is a corner detector that works applying a circle of 16 pixels around the tested corner pixel. If all n contiguous pixels in the circle are brighter or darker than the candidate corner pixel, the candidate is classified as a corner. This algorithm is very

suitable for real-time applications, since its high computational speed, but is not robust to noise and scale changes and depends on a threshold.

- **Hessian detector**: is a blob[1] detector based on the Hessian matrix of the image intensity. This matrix is used to analyze local image structures. Indeed the detector works by searching points where the determinant of the Hessian matrix has a local minima.

Multi Scale Detectors

Multi scale detectors are capable of extracting distinctive features reliably under scale changes. The most used are:

- **LoG**: Laplacian of Gaussian is a blob detector, that automatically selects the scale for different blob sizes. It is invariant to rotation and it produces a good estimation of the characteristics scale for other local structures (corners, edges). LoG detector can be applied for finding the characteristic scale of a given image location or for detect also scale invariant regions. More details in [19].

- **DoG**: Difference of Gaussian is an algorithm that detects blobs and it is based on local 3D extrema in the scale-space pyramid built with Difference-of-Gaussian(DoG) filters. Is faster than LoG, since the computation of Laplacian operators is time consuming, but it is less stable and more sensitive to noise and small changes. Complete description of DoG can be found in [20].

Affine Invariant Detectors

Affine transformation is defined in [17] as any linear mapping that preserves collinearity and ratios of distances. In this sense, affine indicates a special class of projective transformations that do not move any object from the affine space R^3 to the plane at infinity or conversely. Previous discussed detectors can handle only specific affine transformations such as uniforming scaling. If the scale is not uniform in all the directions the scale invariant detector can fail. An affine invariant detector is a generalized version of the scale invariant detector. A more detailed discussion about these detectors can be found in [17].

[1] blob is a region of the image where some properties are constant or approximately constant

3.3.2 Feature Descriptors

The second step is to describe the features and their neighbourhood. Feature descriptors encode these informations in a vector (called descriptor) in order to be matched. Most used feature descriptors are:

- **SIFT**: Scale Invariant Feature Transform, described in [20], is an algorithm where interst points (features) are detected using DoG, chosen as local extrema of Difference of Gaussian, and for each point a feature vector is extracted. Local orientation of the image is estimated, giving invariance to rotation. For each feature a descriptor is computed, based on local image information. The descriptor builds an histogram of gradient orientations of sample points in a region around the keypoint, finds the highest orientation value and the close ones. This will be the dominant orientation of the info-point. Around each keypoint, the description phase of the algorithm, creates a $16x16$ pixel region and samples the image gradient magnitudes and orientations, creating a set of orientation histograms. Each histogram contains samples from a $4x4$ sub-region of the original neighborhood region and has eight orientation bins in each. A Gaussian weighting function it is used to assign weights to the magnitude of each sample point and gives higher weights to gradients closer to the center of the region. The descriptor vector is composed by the values of all the orientation histograms entries, 128 entries for each key-point. Finally, the feature vector is normalized to the unit length in order to gain invariance to affine changes in illumination. To prevent from non-linear illumination changes, a threshold of the values of the vector to a maximum of 0.2 is applied followed by another normalization. The major advantages of SIFT are that orientation and scale do not cause radical changes in the feature vector and the representation is resilent to deformation, such as the one generated by perspective effects. These characteristics are evidenced in excellent matching performance against competing algorithms under different scales, rotations and lighting. The drawback of SIFT is its high dimensionality that causes long computation time.

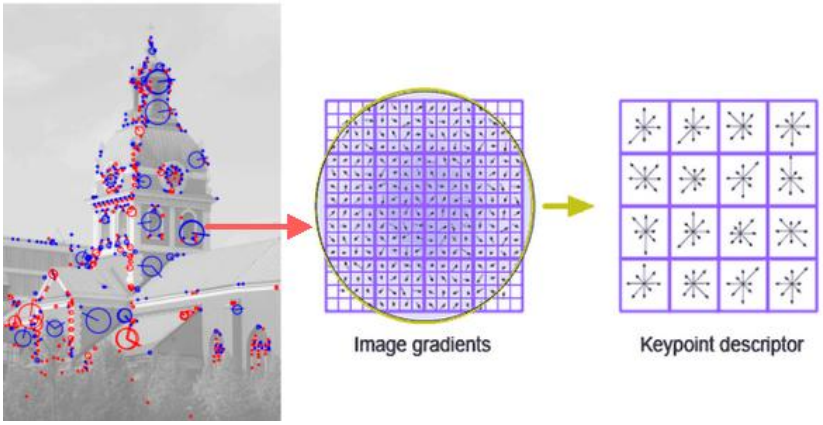

Figure 3.3: SIFT at work: image's detected features are represented, $16x16$ pixel region around one keypoint and the $4x4$ descriptor array, credits [21]

- **SURF**: Speeded Up Robust Features Descriptor is a blob detector and descriptor. The detection phase consists in the usage of two dimensional box filters ($9x9$) where the detector is based on the determinant of the Hessian matrix for scale selection and location. These filters are an approximations of a Gaussian, and represent the lowest scale for computing the blob response maps. The approximated determinant of the Hessian represents the blob response in the image, and it is stored in the blob response map. From this map, the local maxima is detected and the interest points are found applying a non-maximum suppression in a $3x3$ neighborhood, in order to detect steady key points. In order to get the descriptor for each key point, SURF constructs a square region (of dimension 20 times the scale in which the feature was found) centered around the detected interest point and oriented along its main orientation. Then the region is divided in $4x4$ sub-regions and for each one the Harr wavelet [2] responses in the vertical and horizontal directions are computed. The responses are then weighted with a Gaussian window centered at the interest point, in order to increase the robustness against geometric deformations and localization errors. The feature descriptor is normalized to a unit vector in order to prevent illumination changes and has a length of 64 dimensions. This is one of the advantages of SURF, its low dimensionality results in a lower computation time, outperforming SIFT. However, it is not as robust as SIFT, in terms of invariance to rotation. Complete description of SURF is avaible in [22]

[2]Haar wavelet is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis

- **BRIEF**: Binary Robust Independent Elementary Features is a low bitrate descriptor. It belongs to the family of binary descriptors, indeed it uses binary strings instead of vectors of floating point numbers for build the descriptor. It compares the intensity between two pixel positions located around the detected interest points. The BRIEF algorithm relies on a relatively small number of intensity difference tests to represent an image patch as a binary string. This allows to obtain a representative description at very low computational cost. On the other side, BRIEF descriptor is not robust against rotation larger than 35°. Complete discussion of this descriptor can be found in [23].
- **ORB**: Oriented FAST and Rotated BRIEF is a binary descriptor that increases BRIEF performance, dealing with the problem of invariance to rotation. It uses an improved FAST as feature detector, that consists in usage of the Harris corners measure to rank the detected FAST corners, finding the n best ones. Rotation invariance is obtained by computing the orientation based on the intensity centroid. A modified BRIEF is then used as feature descriptor. Indeed, tests are not performed randomly as in the basic BRIEF implementation, but a selection test is first done to learn good binary features. Then a search is performed over all binary tests to find the ones with variance higher than 0.5. A more detailed explanation of this algorithm, with mathematical details is illustrated in [24]. ORB combines the computation speed of BRIEF gaining the invariance to rotation and to scale.

3.3.3 Feature Matching

Feature matching is the task of establishing correspondences between two images of the same scene/object. Once the features are extracted from two images and their detector is computed, a distance function to compare two descriptors must be defined. Then for all the features in the first image, the correspondent features in the second image are found using minimum distance criteria. Given two interest points (one from the first image p and one from the second q), a match between p and q is accepted only if p is the best match for q in relation to all the other points in the first image and q is the best match for p in relation to all the other points in the second image. The optimal distance function and its parameters depend on the data set characteristics, one of the most used is the Euclidean norm distance. The ratio test is used to discard matching candidates for which the correspondence may be regarded as ambiguous. It is defined as the calculation of the ratio of distances of the descriptor in the first image to the nearest and the next nearest image descriptor in the second image. One of this two ratios will be lower than a threshold and the corresponding image descriptor will be the good matching feature.

Figure 3.4: Example of feature matching between two similar images, credits [25]

3.3.4 Key point matching: KLT

Kanade-Lucas-Tomasi(KLT) is a feature tracker algorithm. The KLT tracker basically looks around every feature to be tracked, and uses this local information to find the feature in the next image. Good features are extracted using the Shi-Tomasi corner method, previously described. Then these features are tracked using a Newton-Raphson method for minimizing the difference between past and current frames. An accurate description of this algorithm can be found in [26]

3.3.5 Outlier remove: RANSAC

Outlier removal is a critical step that must be done to avoid false matching that can occour in the feature matching step. Indeed, in enviroments with similar patterns (such a building with many windows) this phenomena can appen. RANdom SAmple Consensus is an algorithm designed for this scope. It consists in a resampling technique that generates candidate solutions by using the minimum number observations (data points) required to estimate the underlying model parameters. The algorithm is divided in three steps:

1. Select randomly the minimum number of points required to determine the model parameters, from the matching step model.

2. Generate new model parameters from these sampled points.

3. Give to the new model a score based on the number of inliers within a pre-set threshold value.

These steps are iterated until the number of inliers is the highest possible, with a maximum of N iterations. The number of iterations necessary for efficient outlier removal depends on the outlier/data points ratio and complexity of the original model. A more detailed description of RANSAC algorithm can be found in [27].

3.3.6 Motion estimation

Figure 3.5 illustrates an example of the visual odometry problem: the relative poses T_{nm} between cameras viewing the same 3D point are computed by matching the corresponding points in the 2D image. If the points' 3D location is known, a 3D to 3D or 3D to 2D method may be used. The global poses C_n are computed by concatenating the relative transformations with respect to a reference frame (can be set to the initial frame). Motion estimation can be done with three different techniques: 3D to 3D, 3D to 2D and 2D to 2D methods.

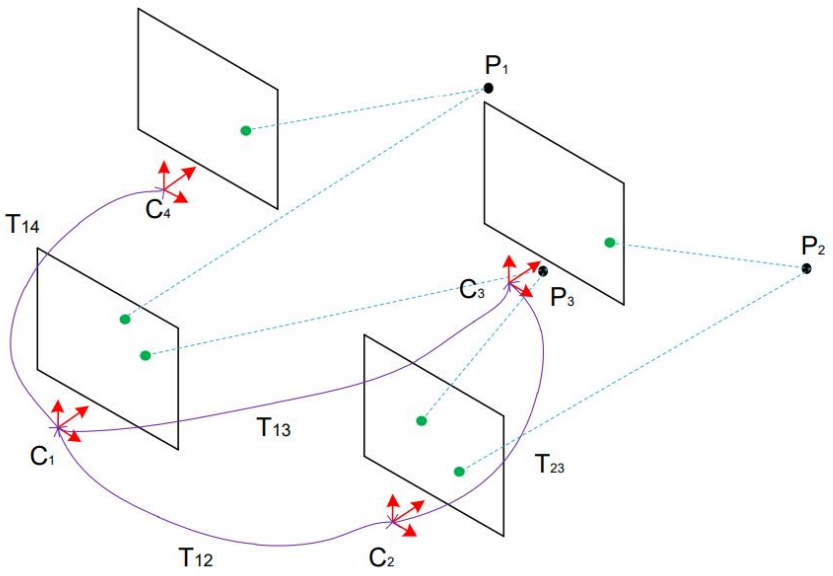

Figure 3.5: Example of a monocular VO system, credits [28]

- **3D to 3D method**: the motion is estimated by triangulating 3D feature points observed in a sequence of images. The transformation between the camera frames is then estimated by minimizing the 3D Euclidean distance between the corresponding 3D points, defined as cost function.

- **3D to 2D method**: This method is similar to the previous approach but the 2D re-projection error is minimized to find the required transformation.
- **2D to 2D method**: This method is used when 3D data is not avaiable, for instance to estimate the relative transformation between the first two calibrated monocular frames where points have not been triangulated yet. Epipolar geometry is used in this case to estimate the transformation between two frames. An example of epipolar geometry problem is illustrated in figure 3.6. The figure shows two cameras, separated by a rotation and a translation, viewing the same 3D point. Each camera captures a 2D image of a 3D. The conversion from 3D to 2D is called perspective projection and it is described in details in [28]. Rotation and translation matrices between two consecutive frames can be extracted using the epipolar constraint described as follows:

$$q' E q = 0 \tag{3.1}$$

where q and q' are the corresponding homogeneous image points in two consecutive frames and E is the essential matrix defined as follows:

$$E = [t]_X R \tag{3.2}$$

where R is the rotation matrix and t is the translation matrix given by:

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.3}$$

and $[t]_X$ is the skew symmetric matrix defined as:

$$[t]_X = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_y \\ -t_y & t_x & 0 \end{bmatrix} \tag{3.4}$$

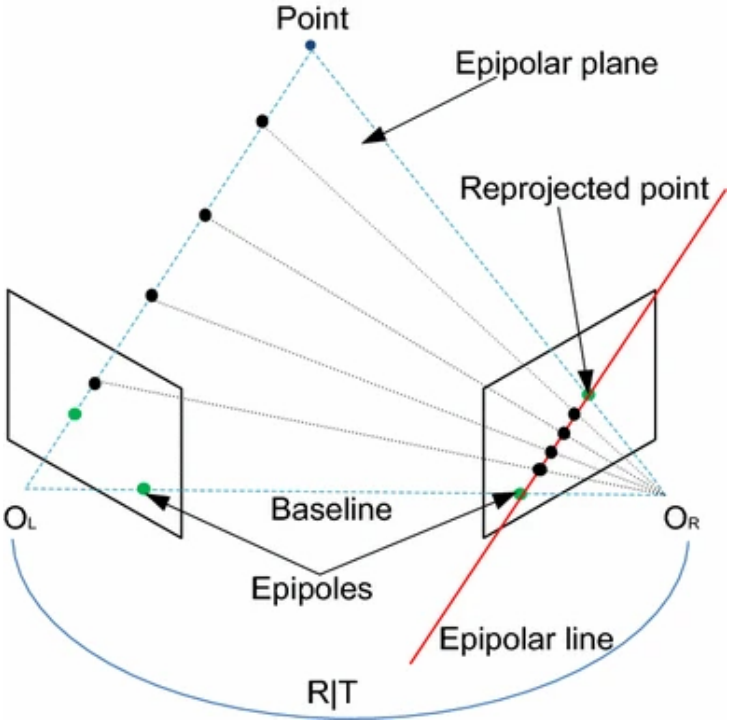

Figure 3.6: Example of epipolar geometry, credits [28]

3.4 Sensor fusion with Kalman Filter

Kalman filter is an algorithm that provides estimates of some unknown variables given the measurements observed over time. It has relatively simple form and requires small computational power. It is used to estimate states based on linear dynamical systems in state space format. The Extended Kalman Filter is an extension of the Kalman filter able to deal with nonlinear models in an efficient way. It linearizes the model in a current estimate. General model's equations for state transition and measurement are:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{3.5}$$

$$z_k = h(x_k) + v_k \tag{3.6}$$

Where f is the function of the previous state x_{k-1}, and of the control input u_{k-1} and provides the current state x_k. h is the measurement function that relates the current state x_k with the measurement z_k. w_{k-1} and v_{k-1} are Gaussian noises for the process model and the measurement model with covariance Q and R. M is the input's model covariance. For the linearization is needed to obtain the Jacobian matrix of each model in each time step:

$$F_k = \mathbb{J}^f_{x_{k-1}} \tag{3.7}$$

$$V_k = \mathbb{J}^f_{u_k} \tag{3.8}$$

$$H_k = \mathbb{J}^f_{h_k} \tag{3.9}$$

The prediction and updtate steps are similar as the linear Kalman filter and are expressed by these equations:

- Predict state estimate:
$$\hat{x_k} = f(\hat{x_{k-1}}, u_{k-1}) \tag{3.10}$$
- Predicted error covariance:
$$P_k = F_{k-1} P_{k-1} F^T_{k-1} + V_k M V^T_k Q \tag{3.11}$$
- Measurement residual:
$$\tilde{y_k} = z_k - h(\hat{x_k}) \tag{3.12}$$
- Kalman gain:
$$K_k = P_k H^T_k (R + H_k P_k H^T_k)^{-1} \tag{3.13}$$

- Update state estimate:

$$\hat{x_k} = \hat{x_k} + K_k \tilde{y_k} \tag{3.14}$$

- Update error covariance:

$$P_k = (I - K_k H_k) P_k \tag{3.15}$$

A more detailed description of Kalman filter and its extended version, with some examples of applications can be found in [29].

Chapter 4

Sensors characterization

In this section it is explained in details the work done on the characterization of the two principal sensors used in this project: the digital camera and the IMU. Since the IMU used was a low-cost one, its characterization was very important to know how much is reliable, compared to the high cost ones.

4.1 Camera calibration

The calibration of the camera step was needed to retrieve its intrinsic parameters and then remove distortion effects from each frame.
Distortion is a phenomena caused by camera's lenses and can be of two kinds: radial distortion or tangential distortion. Radial distortion causes straight lines to appear curved and becomes larger the farther points are from the center of the image. The correction of a distorted image can be seen in figure 4.1.

Figure 4.1: Effect of the distortion correction

Tangential distortion occurs because the image-taking lens is not aligned perfectly parallel to the imaging plane, as a consequence some areas of the image may look

nearer than expected.
From camera calibration the intrinsic matrix K can be retrieved, defined as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

where f_x and f_y are focal length and c_x with c_y represent the optical centers. This matrix K is necessary in the perspective projection step, to transform 3D world's coordinates into 2D camera's plane, as described in details in [28].
Calibration step was done using a pre-existent script of OpenCV and, to verify result's accuracy it was re-done with Matlab's tool. In order to retrieve camera's intrinsic parameters is needed to have a test pattern with known dimensions. An example of acceptable one is a chessboard. For that reason was created a dataset, acquiring images (with the ROV's camera) of a chessboard that has 7 rows and 10 columns with square's size of 10cm. The ROV was moved to get different perspectives of the chessboard, in this way the accuracy of the calibration increases. 54 images were used to retrieve K matrix. An example of calibration frame is illustrated in figure 4.2.

Figure 4.2: Chessboard used for the camera's calibration

4.2 IMU characterization

The IMU used for this work was the MPU-6050. Combines a 3-axis gyroscope and a 3-axis accelerometer and communicates through I2C protocol [1]. It is commonly used in many projects since its reduced cost, availability and easiness of use.

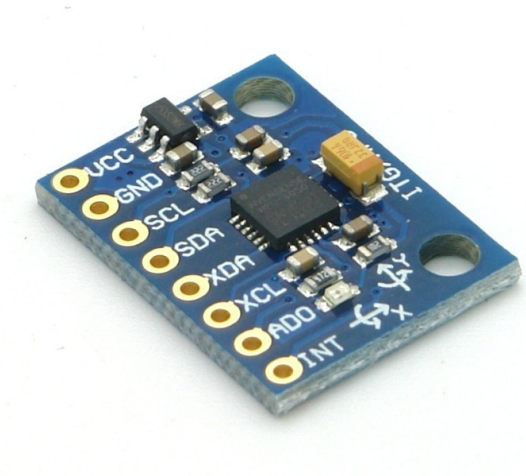

Figure 4.3: IMU MPU-6050

4.2.1 Setting for data acquisition

The acquisition IMU's data was implemented using the Rasberry Pi, connecting them toghether through I2C. Both were placed inside the waterproof platform constructed for the underwater tests. The description of this structure is presented in 5.4.1.
A library for retrieve IMU's readings was developed in Python coding language. This step was necessary in order to exploit all the MPU-6050 characteristics. Indeed, it has configurable registers for set the sampling rate (that can be up to $8KHz$), enable a low pass filter (and set the frequency of this filter), select gyroscope and accelerometer ranges. Acquisition rate was set to $200Hz$ and low-pass filter was enabled with a value set to $10Hz$. Sample rate was decided with initial acquisitions. It was noticed that the readings at higher frequencies were very noisy, for that reason it was also decided to use the internal low-pass filter and set it to the lowest frequency possible. The effects of these choices on noise removal can be seen in figure 4.4.

[1] I2C is a serial communication bus widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance

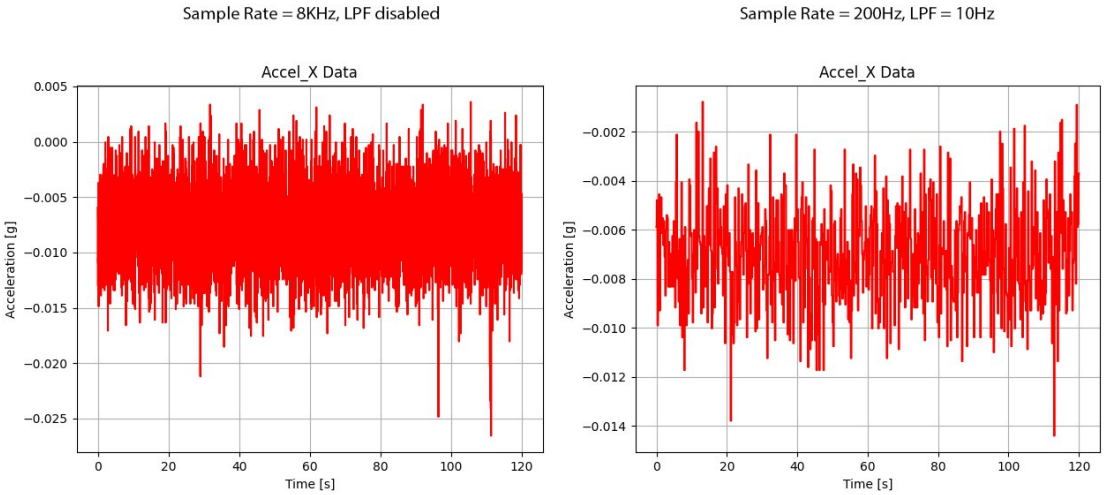

Figure 4.4: Comparison between original raw data (left) and using a Low pass filter (right)

4.2.2 Calibration

Calibration step of an IMU is crucial to have reliable readings in its application. High cost IMUs come with fabric calibration values, that are stored in the firmware or inside the IC's memory, providing accurate measurements off the shelf. Low cost IMUs, such the one used in this study, are usually poorly calibrated.
Calibration process consists in the estimation of biases and offsets in steady and known conditions. These data will be used to correct future sensor's reading.
This was achieved developing a script that acquires sequentially the data from the inertial sensor, in fixed positions, and computes the calibration factors, storing finally the values in a text format.
Firstly the offset that affects the gyroscope is measured. Indeed, each one of the three axis of the gyroscope should measure $0°/s$ in steady conditions. Once the platform is fixed in a stable position the script acquires 1000 samples of the gyroscope's readings and computes the mean value for each axis, finding the offset values.
The second part concerns to the calibration of the accelerometer. It requires taking advantage of the acceleration due to gravity, that can be used in the positive and negative orientation of the IMU. Additionally position of the IMU perpendicular to gravity is used in order to acquire a third calibration point. This results in three unique values that can be combined to formulate a linear fit between the three values and the values outputted by each axis of the accelerometer. For each axis the platform was fixed in three position:

1. Axis upward against gravity
2. Axis downward toward gravity
3. Axis perpendicular to gravity

For each axis position the script acquires 500 samples, and after the third orientation computes the values of the intercept and slope that will be used for least-squares fitting of a first-order model for the axis of the accelerometer. Figure 4.2 shows the platform during the calibration phase of the accelerometer, with the Y-axis pointed upward against gravity. The platform was placed in a plane surface, and each orientation of the axes was done with the help of a bubble level, in order to achieve the maximum accuracy possible.

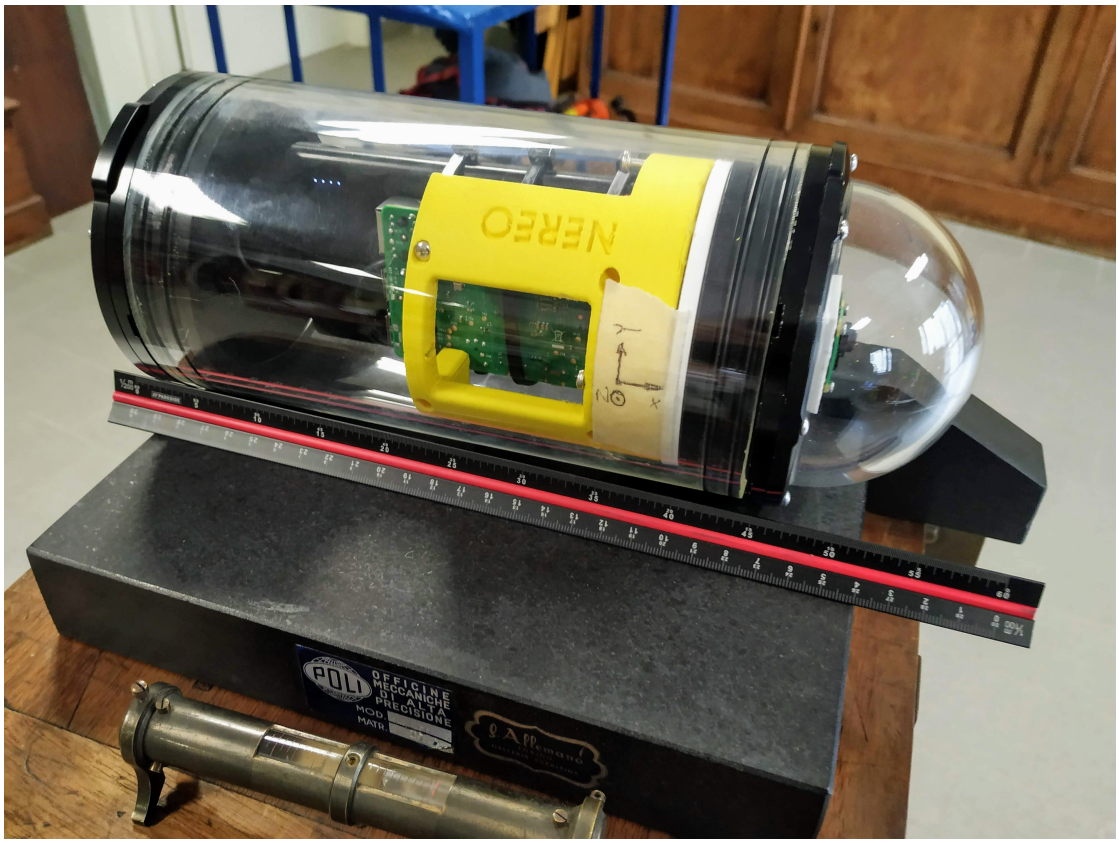

Figure 4.5: Platform during accelerometer's calibration

4.2.3 Noise reduction with wavelets denoising

Wavelet noise reduction has been proven as a useful tool in signal analysis, and it is widely used in denoising applications [30]. Wavelets are local in both frequency/scale (via dilations) and in time (via translations). In many classes of functions can be represented by wavelets in a more compact way with respect Fourier's transformation. The filtering of signals using wavelets is based on the idea that as the Discrete Wavelet Transform (DWT) decomposes the signal into details and approximation parts using a selected wavelet. At some scale the details contain mostly the insignificant noise and can be removed or zeroed out using thresholding without affecting the signal. The signal is then reconstructed with the Inverse Discrete Wavelet Transform (IDWT). More details on theoretical background about Wavelet denoise can be found in [31]. In this work the selected wavelet was the Daubechies 10 (db10), with the number of wavelet levels fixed to 7 and the threshold was assigned to the variance of the signal. The parameters were selected empirically, with hit and trial testing of various mother wavelets and levels. In figure 4.6 is presented the result of this filter in a simple acquisition. It is clear the signal is smoother, and does not lose important information.

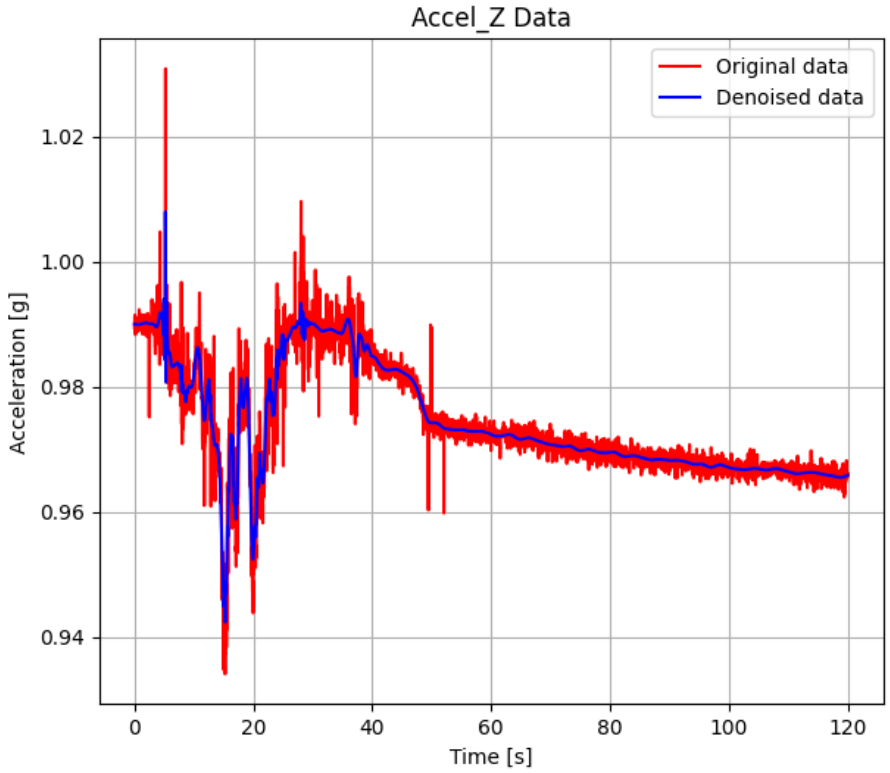

Figure 4.6: Effect of wavelet denoise process on acquisition

4.2.4 Six faces test

An important aspect of the IMU is the drift of the measurements over the time. Defined also as bias stability, is needed to know how much a measurement is reliable during an acquisition process. This drift can be caused by temperature variation, electromagnetic noises or poor sensor performance. MPU-6050's datasheet gives for the accelerometer a sensitivity change value of $0.02\%/°C$. To see the presence and the value of sensor's drift, the six faces test was implemented. It consists in place the platform in six different positions and perform a long acquisition, while the IMU stands still. For each of the axis, the platform was moved in order to oriented the axis downward toward gravity and then upward against gravity, as can be seen in figure 4.7.

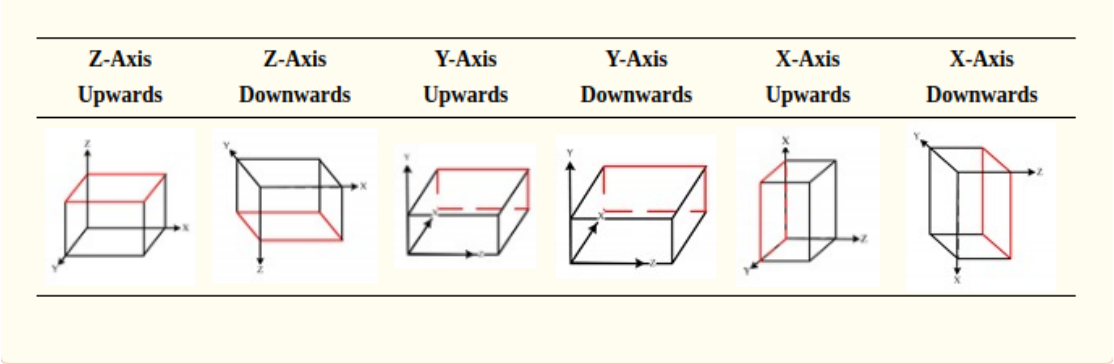

Figure 4.7: Orientations of the platform during the six faces test

The results were stored in a text file and then plotted. Acquisition time was fixed to 45 minutes. The focus of this test was not only to see drifts but also to verify if these drifts are different with the orientation of the IMU. In figure 4.8 is illustrated the two different orientations on the Z axis. For clearness of the graph, absolute value of the readings was plotted.
Accelerometer's data is not particularly affected by drifts for the axes perpendicular to the gravity vector. Looking at the accelerometer data for Z, instead, it can be noticed a small drift over the time, that has opposite slope for the two orientations. This drift can be considered negligible and most probably caused by the increase of the temperature inside the IMU's housing.
The readings from the axis of the accelerometers show offsets that cannot be ignored: X and Y readings, in the test illustrated in figure 4.8, should be equal to 0 since they are perpendicular to the gravity vector. These offsets can cause errors in the navigation estimation, and for that reason they were taken into account and subtracted from the measures in the underwater test done in 5.4. For the test were considered the offsets measured with the Z axis pointing upwards, since it was the orientation of the platform used. Table 4.1 shows the offsets of the accelerometer's readings. Values of the X axis have the highest standard deviation, meaning that the removal of the offset will be not accurate, since this offset is not constant during the readings.
Gyroscope drifts over the axes is more marked, even can be considered acceptable. For X axis and Y axis values the trend is pretty the same for the two different orientations. Z axis values shows instead opposite trend of the drifts between the two orientations. Table 4.2 shows the values for the offsets of the gyroscope.
Same observation were made for the other axes orientations, since data showed that drift's slope is correlated with the axes orientation. For a better characterization of drift the next step done was to compute the Allan variance.

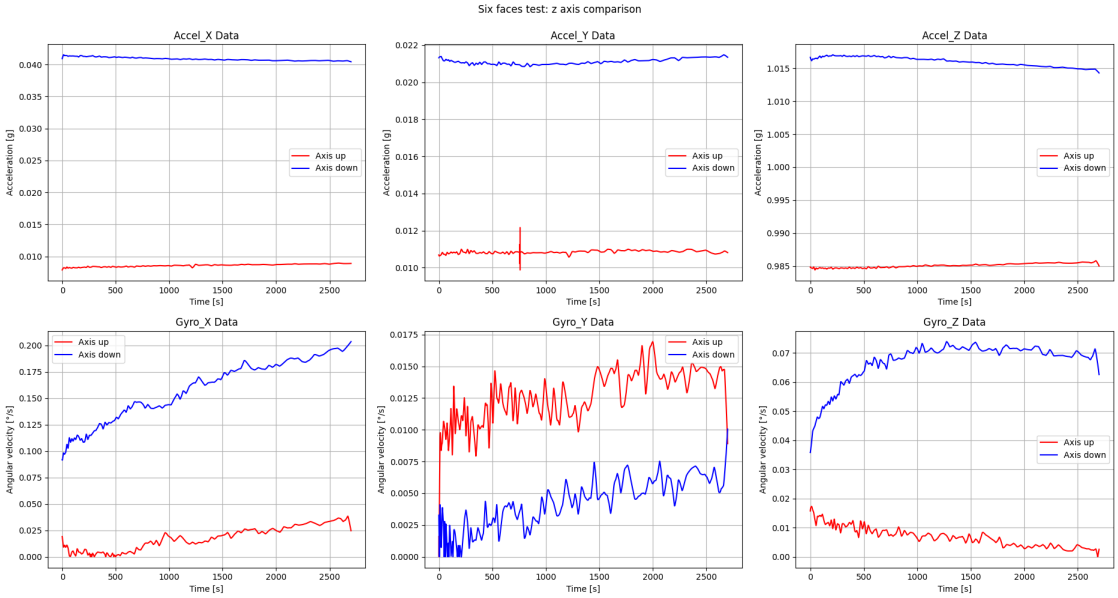

Figure 4.8: Six faces test: Z axis

Accelerometer Axis	Average offset $[m/s^2]$	Standar deviation $[m/s^2]$	Min offset $[m/s^2]$	Max offset $[m/s^2]$
X axis	8.37e−2	2.04e−3	7.74e−2	8.79e−2
Y axis	1.06e−1	7.31e−4	9.61e−2	1.20e−1
Z axis	1.47e−1	2.97e−3	1.39e−1	1.53e−1

Table 4.1: Six faces test with Z axis upwards: accelerometer offsets.

Gyroscope Axis	Average offset $[°/s]$	Standar deviation $[°/s]$	Min offset $[°/s]$	Max offset $[°/s]$
X axis	1.41e−2	1.05e−2	1.89e−7	3.84e−2
Y axis	1.21e−2	2.06e−3	1.23e−6	1.69e−2
Z axis	7.09e−3	3.50e−3	1.53e−7	1.72e−2

Table 4.2: Six faces test with Z axis upwards: gyroscope offsets.

4.2.5 Allan variance

The Allan variance is a time-domain analysis technique originally developed to study the frequency stability of precision oscillators. Being a directly measurable quantity, it can provide information on the types and magnitude of various noise terms. Because of the close analogies to inertial sensors, this method has been adapted to random-drift characterization of a variety of devices. An accurate description of Allan variance and IMUs characterization process using this method, can be found in [32]. From Allan variance analysis the covariance values for each IMU's error can be extracted and used in the final implementation. Figure 4.9 shows the hypothetical Allan variance of an inertial sensor. The five basic noise terms are: quantization noise, angle random walk, bias instability, rate random walk, and rate ramp. The quantization noise is one of the errors introduced into an analog signal by encoding it in digital form. Bias instability error is originated by the electronics or other components that are susceptible to random flickering. Rate random walk is a random process of uncertain origin.

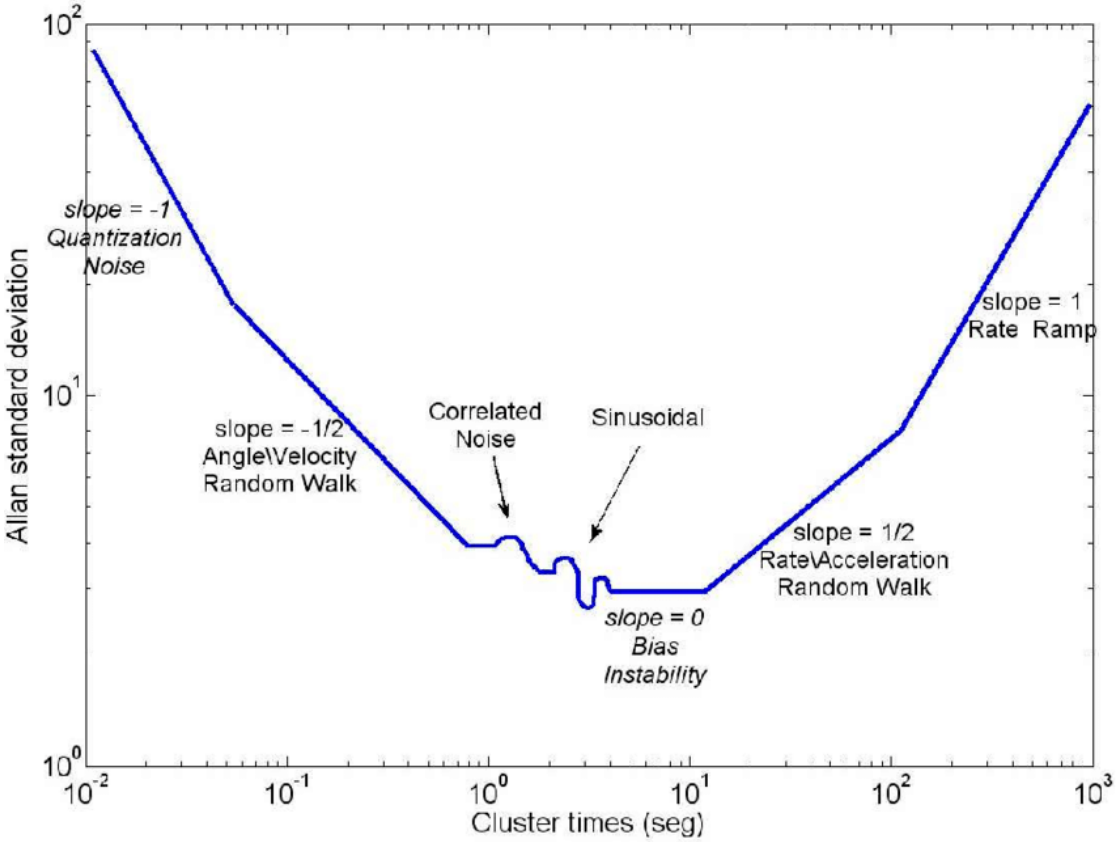

Figure 4.9: Ideal Allan variance for an inertial sensor, credits [33]

The data acquired for the six faces test was used to compute the Allan variance. The variance was computed only for the accelormeters, and it is plotted in figure 4.10 First row in the figure shows for each of the three different orientations (X,Y,Z axis upwards toward gravity) the different values of the variance computed for X-axis, Y-axis, Z-axis accelerometer's data. The second row shows instead the X-axis, Y-axis, Z-axis accelerometer's computed variance over the three different orientations. This is useful to see how different axis's orientation affect the computation. From the results it can be noticed that the quantization noise affects the accelerometer's axes with different values. X axis accelerometer has the same errors for the different rotations whilst the random walk error of the Y axis readings are influenced by the change of orientation. Z axis accelerometer's bias instability seems to be affected by the orientation of the Y axis. In table 4.3 are shown the values for random walk and bias instability, relative to the IMU having the Z-axis pointing towards gravity vector.

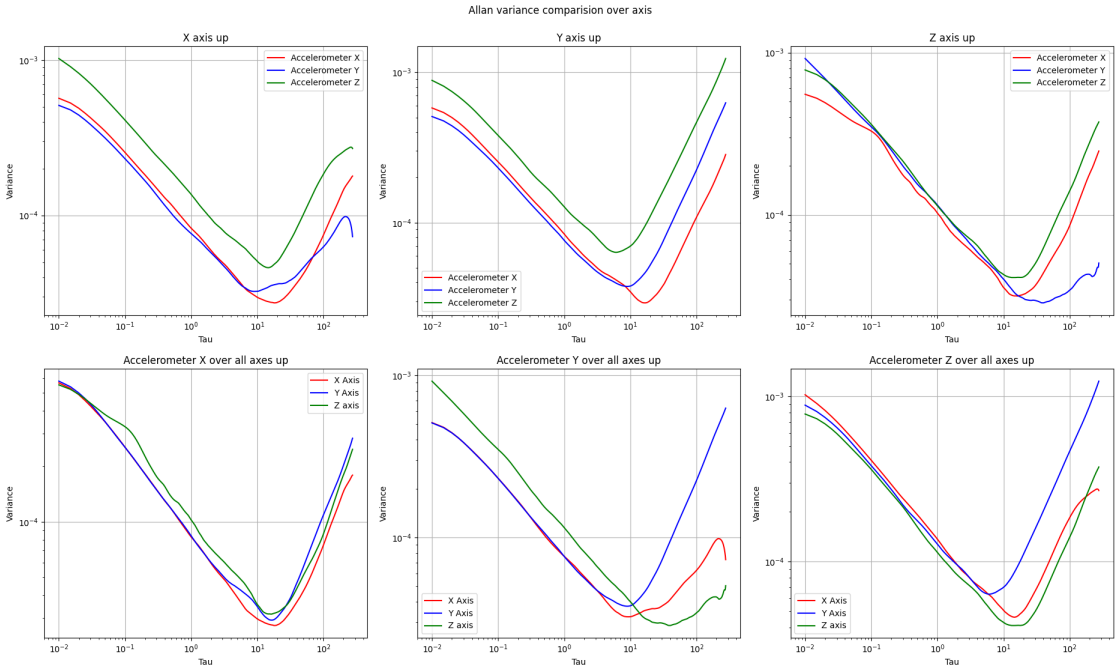

Figure 4.10: Allan variance results for MPU6050

Accelerometer Axis	Random walk [m/$s^2/\sqrt{s}$]	Bias instability[m/s^2]
X axis	9.66e−5	3.16e−5
Y axis	1.14e−4	2.96e−5
Z axis	1.13e−4	4.10e−5

Table 4.3: Allan variance values for Z axis upwards.

Chapter 5

Data Integration and Positioning

In this chapter the integration of the experimental work is presented. Firstly is explained in details the algorithm designed for this work, followed by the steps done to derive that. These steps were made by test done on the different parts of the system. The first test presented was done on a pre-existing underwater dataset, focused on the comparison of the feature detectors.
Then, the Visual Odometry algorithm is described: its design, the integration with the EKF (see 5.1.5 and 5.3.3) and the tests done, describing also on the data acquisition done for these tests. Lastly is presented the work done in order to test the system in an underwater environment: the creation of a waterproof acquisition unit, the setup of the test and its results.

5.1 Algorithm

The algorithm proposed for this study is represented schematically in figure 5.1. In the following sections are explained in details the various blocks that compose the implementation. This application was tested with datasets acquired in and underwater environment. Details about the data acquisiton and test's results are described in 5.4.

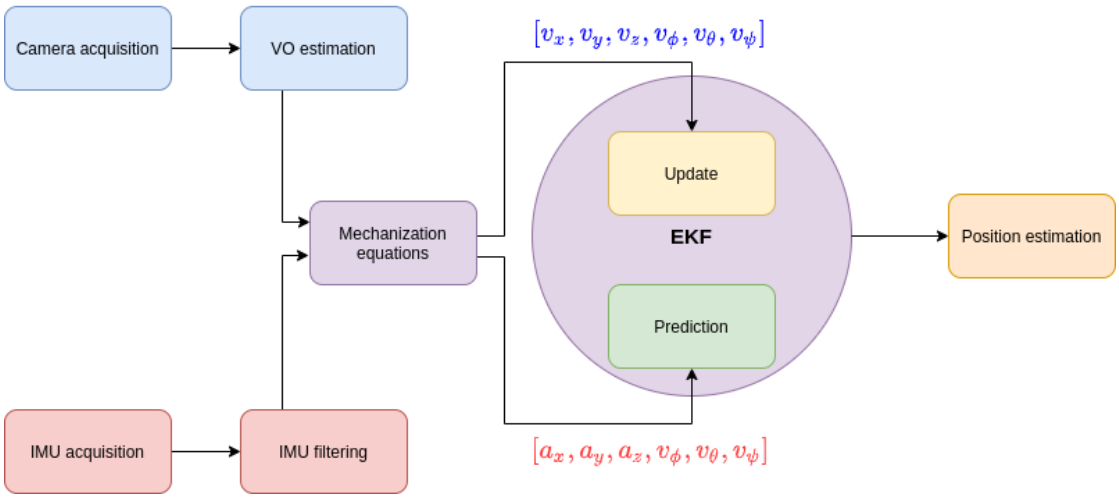

Figure 5.1: Proposed implementation of the complete project's algorithm

Where v_x, v_y and v_z are the velocities alongside the respective axes of the body, a_x, a_y and a_z the accelerations and v_ϕ is the angular velocity around X-axis,v_θ around Y-axis,v_ψ around Z-axis.

5.1.1 Visual Odometry

Visual Odometry section of this implementation was derived from previously work done that is described in 5.3.2. The VO sampling to camera rate was increased to $2.5Hz$. This choice was made to increase VO's performance. The output response of the VO was modified, in a way that for frame it returns linear velocity trough axes and also angular velocity.
Therefore the output vector is composed as : $[v_x, v_y, v_z, v_\phi, v_\theta, v_\psi]$.
Inclinations around the axes are defined as the heading of the moving object. They are commonly named as: roll, pitch and yaw. Figure 5.2 shows the convention adopted.

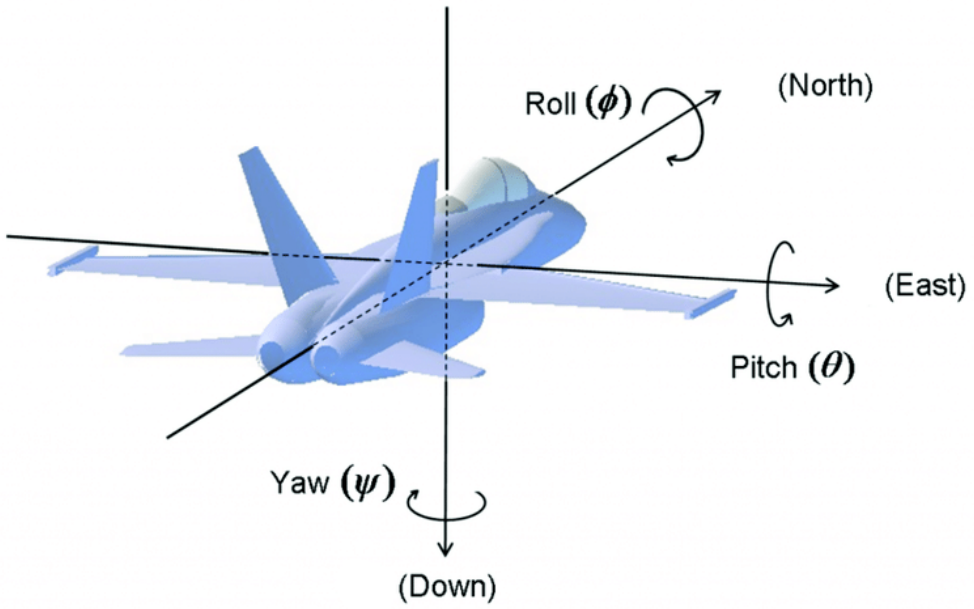

Figure 5.2: Roll, pitch and yaw convention, credits: [34]

5.1.2 IMU

IMU's readings are filtered, as previously described, using the wavelets and, for each axis of the accelerometer is subtracted its mean noise value derived from the six faces test. In this case were considered the offsets measured with the IMU's Z axis pointing upwards, since it was the orientation of the system during the test. Gravity was removed by subtracting $9.81 m/s^2$ from the Z-axis accelerometer data.

5.1.3 Reference frame

Camera and IMU, placed inside the underwater platform (described in 5.4.1), have different orientations within each other. VO reference frame is also different with respect the one selected for the body (the platform constructed for the test in 5.4.1), as it is represented in figure 5.3

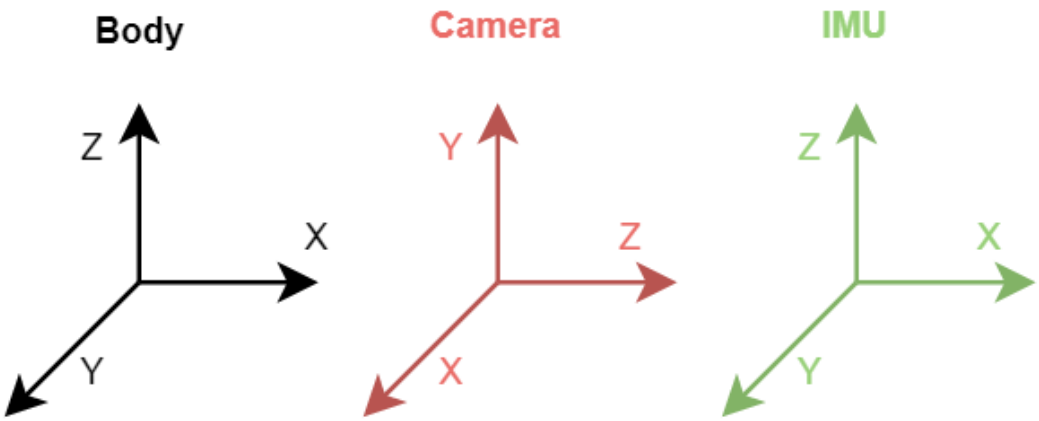

Figure 5.3: Body's, camera's and IMU's reference frame

Inputs from VO were then converted from their frame of reference to the body's one using the following equations:

$$\begin{bmatrix} X_{body} \\ Y_{body} \\ Z_{body} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \end{bmatrix} \tag{5.1}$$

IMU's inputs are not converted since they have the same orientations of the body's reference frame.

5.1.4 Mechanization equations

Values in body's frame must be translated to the global frame. This is caused by the rotation of the body with respect the global reference frame. Mechanization equations were used to perform this translation. Euler angles used in the mechanization equations describe the orientation of the body frame relative to the global frame, as it can be seen from figure 5.4.

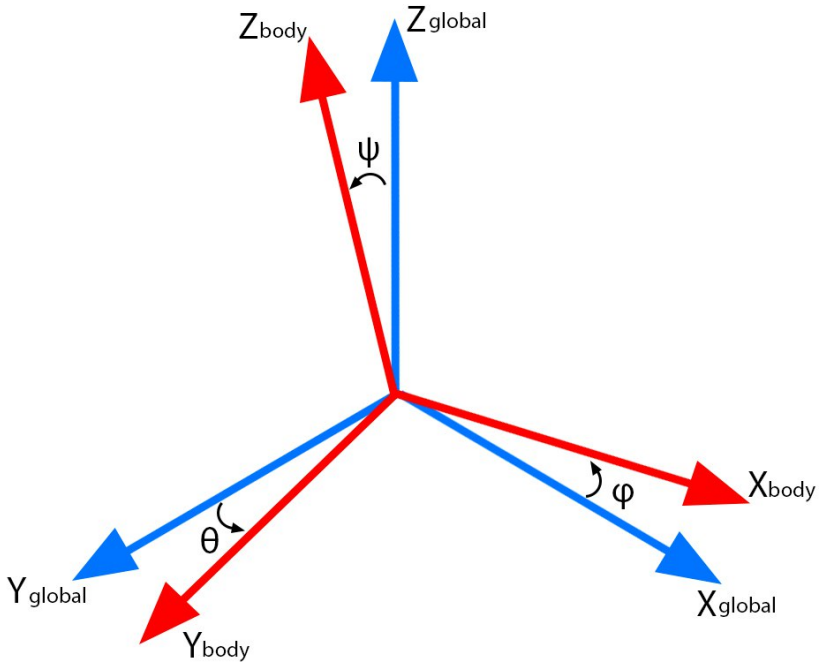

Figure 5.4: Rotation of the body's frame with respect the global's one

The first conversion that is done regards the angular velocity vector of the body frame that it is translated to the global frame with the following equation:

$$\begin{bmatrix} v_{\phi_{global}} \\ v_{\theta_{global}} \\ v_{\psi_{global}} \end{bmatrix} = \frac{1}{\cos\theta} \begin{bmatrix} 1 & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \times \begin{bmatrix} v_{\phi_{body}} \\ v_{\theta_{body}} \\ v_{\psi_{body}} \end{bmatrix} \tag{5.2}$$

Because the above equation requires the global orientation in order to calculate the global change in orientation, a previous calculation or estimation of heading is required. In the system developed, the predicted heading produced within the Kalman filter was used. Heading computed with the Kalman filter is used also for

the transformation matrix C_b^g as follows:

$$C_b^g = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \cos\phi & \sin\phi \end{bmatrix} \tag{5.3}$$

Accelerations are then converted to the global frame using the previously described matrix, with this following equation:

$$\begin{bmatrix} a_{x_{global}} \\ a_{y_{global}} \\ a_{z_{global}} \end{bmatrix} = C_b^g \times \begin{bmatrix} a_{x_{body}} \\ a_{y_{body}} \\ a_{z_{body}} \end{bmatrix} \tag{5.4}$$

More details about mechanization equations can be found in [35].

5.1.5 Sensor Fusion

Fusion between readings from VO and IMU was performed using an EKF. The state vector of the filter was defined as follows:

$$\mathbf{X} = [x, y, z, v_x, v_y, v_z, a_x, a_y, a_z, \phi, \theta, \psi, v_\phi, v_\theta, v_\psi] \tag{5.5}$$

Every time a reading from the IMU arrives it is passed to the filter as input, composing the following input vector:

$$\mathbf{u} = [a_x, a_y, a_z, v_\phi, v_\theta, v_\psi] \tag{5.6}$$

The dynamic equations derived for this filter are expressed in the following equation. Note that are listed in terms of X-axis but they were extended for the Y and Z axes:

$$\begin{cases} x_t = x_{t-1} + v_{x_{t-1}} \times dt + 0.5 \times a_{x_{t-1}} \times dt^2 \\ v_{x_t} = v_{x_{t-1}} + a_{x_{t-1}} \times dt \\ a_{xt} = u_{x_t} \\ \phi_t = \phi_{t-1} + v_{\phi_{t-1}} \times dt \\ v_{\phi_t} = u_{v_{\phi_t}} \end{cases} \tag{5.7}$$

The difference between two consecutive IMU's readings timestamps are used as dt in the filter. Jacobian's of state update are computed (see 3.7 and 3.8) at each prediction obtaining the prediction covariance matrix 3.15. Status update of the filter is done with readings from VO. Measurement vector is thus composed:

$$\mathbf{z} = [v_x, v_y, v_z, v_\phi, v_\theta, v_\psi] \tag{5.8}$$

Measurement matrix was defined as follows:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

5.2 Comparison of features detectors

In this section is described the first test done on a underwater dataset. The aim of this test was to compare the previously described feature based method, in order to choose the best one for the application of this project. Another interest of this test was the enhancement of the image useful to achieve better performance in the feature detection step. An additional attention was given to the computational time, for the purpose of evaluate the possibility to achieve a near real-time implementation in the complete system.
Performace of feature detector is defined as the number of matches between two consecutive frames. Computational time, instead, as time elapsed for detect, describe and match the features of frame $n+1$ with the ones present in frame n.

5.2.1 Aqualoc dataset

Dataset used in this test was provided by Ferrera M., Creuze V., Moras J., Trouvé-Peloux P.. It consists in a set of sequences recorded in three different scenarios: an harbor at a depth of a few meters, a first archeological site at a depth of 270 meters and a second site at a depth of 380 meters.
Acquisition was done using an ROV equipped with a monochromatic camera, an IMU, a pressure sensor and a embedded computer. Camera acquisition rate was fixed to $20Hz$ whilist IMU was $200Hz$. More details about hardware implementation, camera calibration and dataset description can be found in [36]. The sequence used for this test was the 04 of the second archeological scenario: recorded at a depth of 380 meters, has a duration of 11 minutes and 09 seconds for a travelling distance of 18.1 meters and includes some visual disturbance such as turbidity, back-scattering, sandy clouds and the presence in some frames of the dynamics robotic arm of the ROV.

5.2.2 Implementation

Hardware

The hardware used for this test was a Rapsberry Pi 4.. Raspberry Pi is a low-cost and compact computer that has a quad-core CPU, capable of running at $1.5GHz$, $4GB$ of RAM and a dedicated GPU [37].

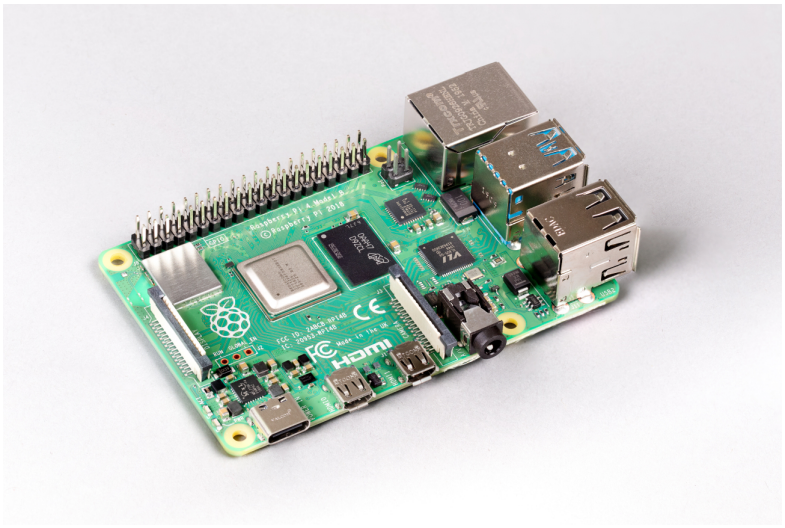

Figure 5.5: Raspberry Pi 4

Software

All the algorithms were implemented using Python as coding language. This allowed to use the OpenCV library, an open-source library which exploits a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms [38].

Pre-process

To improve the performance of the feature detectors a pre-processing step for each image was done. This improvement consisted in a contrast enhancement, using a Contrast-limited Adaptive Histogram Equalization (CLAHE) algorithm. It is an adaptive histogram equalization, image is divided into small blocks called "tiles" and each of these blocks are histogram equalized. In images after CLAHE processing the features are more recognizable. Indeed in figure 5.6 can be seen that the feature detector works better, finding more features, in the contrast enhanced image. A detailed description of CLAHE with an application in facial feature detection can be found in [39].

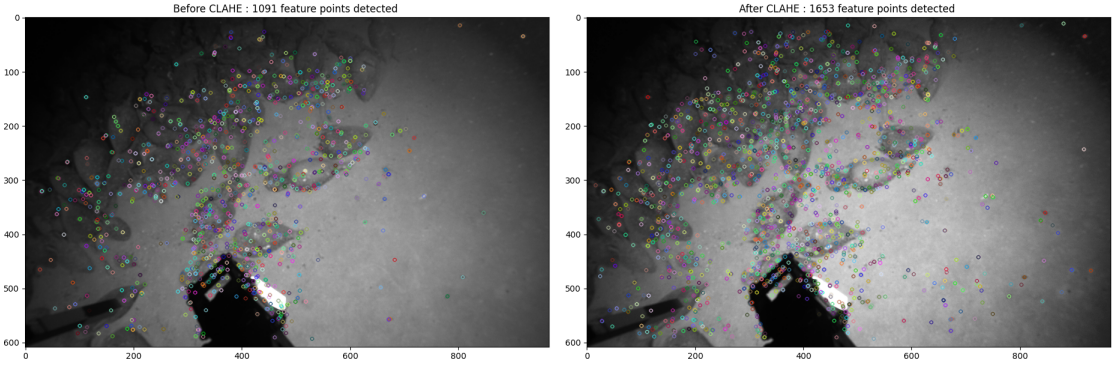

Figure 5.6: Feature detection enhancement with CLAHE (using SURF as feature detector/descriptor)

5.2.3 Results

The test was organized as follows: with a fixed offset n, every n frames in the dataset the script picks the frame f and the $f + n$ frame, applies over the frames the pre-processing step, extracts and describes from them the features for each method. Then it matches the features found in the f frame with the one in $f + n$ frame, computing the matching ratio and the elapsed time. The offset was fixed to 10 because the ROV moves at very low speed and since between two consecutive frames there is a difference of acquisition time of $5ms$ the displacement is almost zero, and so the scenario does not change significantly.
The feature detectors and descriptors compared were:

1. SIFT (Scale-Invariant Feature Transform)
2. SURF (Speeded Up Robust Features)
3. ORB (Oriented FAST and Rotated BRIEF)
4. Shi Tomasi as detector and SIFT as descriptor
5. Shi Tomasi as detector and SURF as descriptor

6. Harris as detector and SIFT as descriptor
7. Harris as detector and SURF as descriptor

The results given by the script were:

1. Matches: features of frame f found in frame $f + n$ (in figure 5.7)
2. Match ratio: ratio of features in the frame f successfully matched in the consecutive frame (in figure 5.7)
3. Feature Detection and Description computational time (in figure 5.8)
4. Pre-Processing time (in figure 5.9)

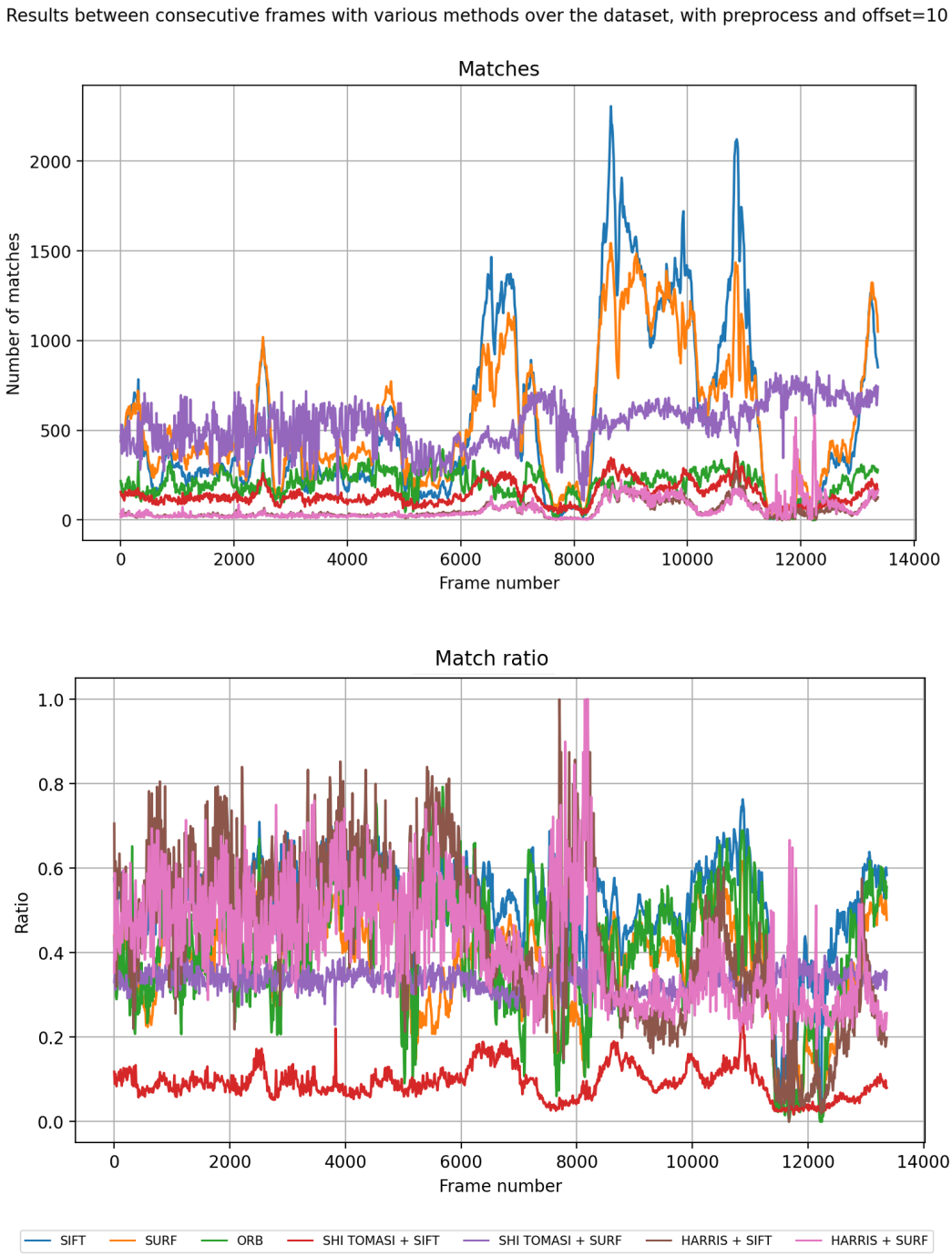

Figure 5.7: Perfomance of various methods over the dataset: number of matches and match ratio

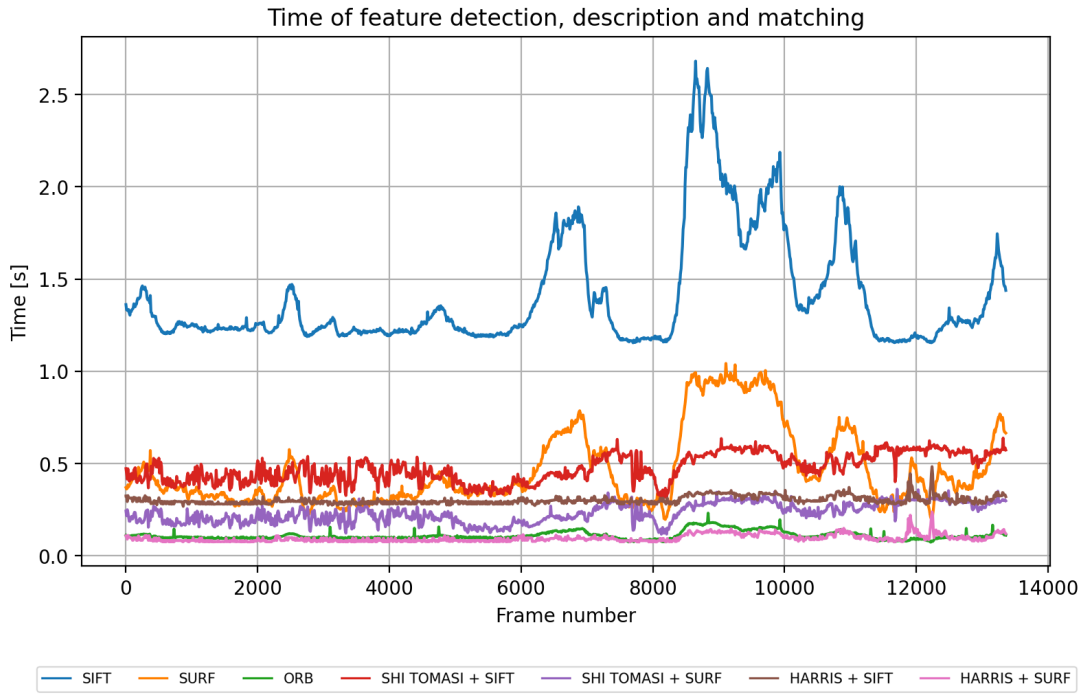

Figure 5.8: Perfomance of various methods over the dataset: computing time for each method

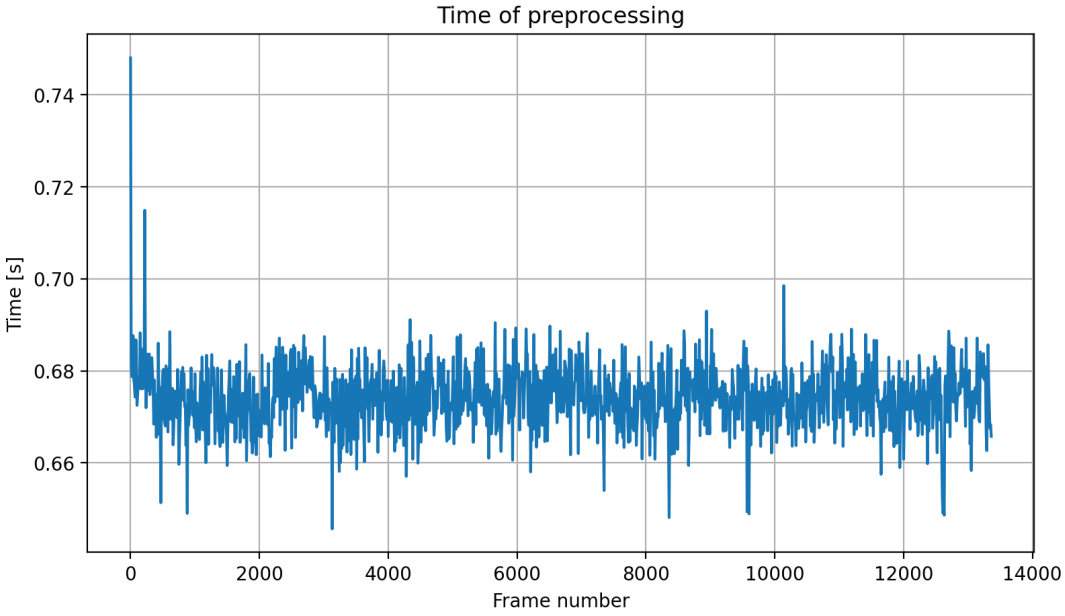

Figure 5.9: Perfomance of various methods over the dataset: pre-process time

From figure 5.7 can be seen that over all the dataset the number of features found increases after frame 8000, for all the methods. This is because the scenario became more rich of elements of interest, such as amphorae.
From figure 5.6 it is clear that the robotic arm gives a large number of features inside the image. To avoid biased results each frame was cropped in order to exclude the robotic arm and focus only on scenario's features. The result shows that the best method for number of matches is SIFT but it has as drawback the highest computational time(looking at figure 5.8. SURF has similar results for detection and matching and has a smaller computational time, but it is still not acceptable for the requirements.
Table 5.1 is useful to have a view in numbers of the performance. Looking at the data of this table, ORB was chosen as method for feature detection and description. Indeed it has the smallest computational time, keeping a good match ratio even if the number of features is not too high. A valid alternative could be Shi Tomasi with SURF even if it is slightly slower than ORB and has a lower match ratio but compensates with a higher number of feature detected.
From figure 5.9 and table 5.1 is clear that the pre-process time affects a lot the computational time of the complete algorithm of this study. Also for this reason the choice fell on ORB.
Pre-process time is the same for each feature detection and description method tested, since it was done before the feature detection step.

Method	Average matches	Max matches	Average match ratio	Max match ratio	Average time [s]	Max time [s]
SIFT	556.75	2306.0	0.517	0.763	1.403	2.681
SURF	551.17	1543.0	0.373	0.616	0.466	1.042
ORB	197.98	394.0	0.404	0.792	0.110	0.231
SHI THOMASI + SIFT	143.53	380.0	0.092	0.245	0.475	0.638
SHI THOMASI + SURF	534.57	828.0	0.334	0.401	0.232	0.361
HARRIS + SIFT	52.81	273.0	0.439	1.0	0.301	0.483
HARRIS + SURF	59.69	584.0	0.419	1.0	0.095	0.237
Pre Process					0.674	0.748

Table 5.1: Results of performance test

5.3 Visual Odometry test

In this section is explained the second test done, which was centralized in VO performance with the selected feature detection and description method. Dataset for this test was created acquiring images in an submerged environment, using a pre-built ROV. The ROV was tracked during the acquisition with a GPS antenna in order to have a ground truth track. The first part of this work was focused on path reconstruction with only VO, paying attention to the robusteness of the VO algorithm and testing various frame rates. In the second part the algorithm was modified introducing an Extended Kallman Filter (EKF) in order to improve VO performance, using GPS data as additional sensor. This was made to test sensors fusion in anticipation of the final step of the project which includes a sensors fusion of monocular VO and IMU data.

5.3.1 Data acquisition

The creation of this dataset was done using an ROV, the *BlueROV2* built by *BlueRobotics* [40]. It is a compact and low-cost ROV, equipped with a monocular camera, a Raspberry Pi and various sensors.
The acquisition site chosen was a park in front of Polytechnic of Turin. It was selected in order to have an ideal dataset, with the possibility to record in a simple way the ground truth track, using a GPS receiver.
The ROV was placed over a cart and moved by hand for the length of the path. The GPS antenna was situated on the left side of the ROV, alined with the center of the camera, as can be seen in figure 5.10. The GPS receiver employed was the *Tersus Precis BX316*, the software used for the acquisition of the video-stream was *QGroundControl*, an open-source program developed by *BlueRobotics*. The acquisition rate of the camera was fixed to $25Hz$, and for each frame acquired its timestamp was stored.
Some markers were put on the asphalt in order to have referring points over the track and to have more features to be detected on the bottom side of the image.

Figure 5.10: BlueROV2 with GPS antenna

Camera was calibrated as illustrated in 4.1, retrieving its intrinsic parameters.

5.3.2 Visual Odometry Algorithm

In this section is described the algorithm used for this test, focusing in details on the Visual Odometry part.
The complete VO sequence is shown in figure 5.11. The first step is to enhance the contrast in the frame, applying CLAHE algorithm as described in 5.2.2.
From the frame features are extracted and described, using ORB as detector and describer. Then, using KLT, the features extracted are compared and tracked with the ones of the previous frame. KLT is used to get the optical flow of the features matched from frame n and frame $n+1$. In this step a backtracking check is done to ensure good features: it consists in tracking the set of features f_n of frame n into the successive frame $n+1$ obtaining the corresponding f_{n+1} features. Then, this set is tracked in the frame n and the set of features f'_n is generated. Then the difference between f_n and f'_n is done and if it is below a certain threshold the features are considered good. Good features are returned for frame n and $n+1$ and, if there are not enough good features for the pair of frames a warning is returned. Indeed, if there are no good features for the pair an error is generated. In addition

of that, this step, returns also the difference between the frames, meant as the distance between the tracked features in the pair.
This distance is used to check if the pair of frames is too similar, indicating that very likely the ROV is stopped. If this condition is verified, the algorithm skips this frame and goes to the successive one.
Otherwise, the next steps are the estimation of the essential matrix E and, from that, the rotation matrix R and the translation matrix t are extracted.
Matrices R and t are used in the triangulation step. A cloud of point is generated from them in order to derive the scaling factor. In details, from the cloud generated by frame n, the distance between two corresponding points is calculated. This distance is then dived by the corresponding points' distance in the point cloud generated by frame $n + 1$, obtaining scaling factor s.
Last stage of the VO algorithm is the one responsible of the path's reconstruction. It concatenates the translation and rotation matrices, applying the scale factor to the translation matrix.

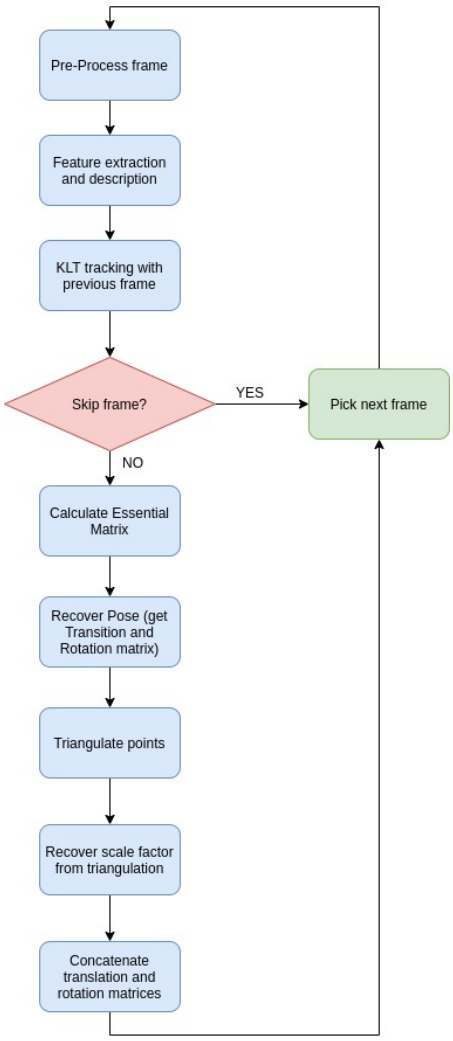

Figure 5.11: Visual Odometry algorithm

This sequence is iterated through all frames with a downscale factor of the camera frame rate fixed to 20. This choice was made in order to gain robustness against fake detected movements caused by rapid changes in the scenario (for example a pedestrian that passes in front of the camera). Another advantage of this choice is the reduced computational time. With this criteria the update rate of this algorithm was $1.25Hz$, an acceptable choice considering ROV's slow movement.

5.3.3 Sensors Fusion

In this second part the outputs of VO algorithm and the measurements given by GPS were fused using an Extended Kalman Filter (EKF) in order to achieve a more reliable motion's estimation. In figure 5.12 is shown how was made.

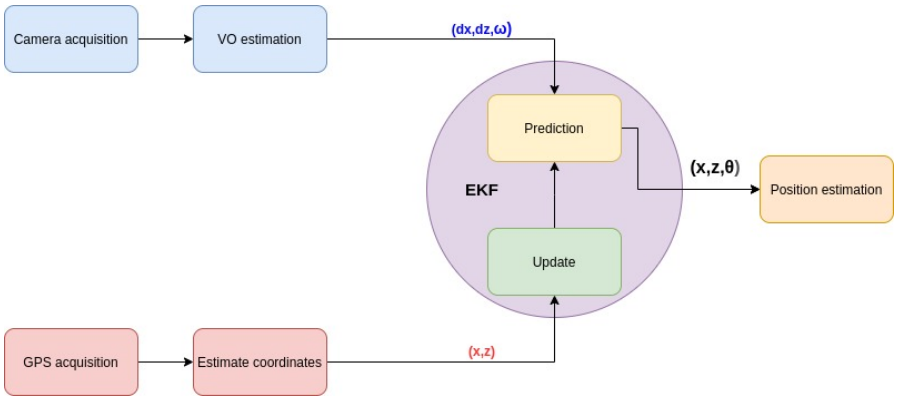

Figure 5.12: Sensors fusion using an EKF

Model

For this system was considered as center of the ROV the center of the camera, since its negligible dimensions. The GPS antenna was placed at the same height and was distanced from the camera center by $29.5cm$. In order to simplify the computation this difference between the two centers was considered equal to 0. Since the ROV was moved upon a cart, the displacement was tracked only in two dimensions.
The cartesian diagram of this system is shown in figure 5.13. In this figure X_w and Z_w are world's coordinates and correspond to northing and easting, respectively. The ROV is represented in world's coordinates by x_t and z_t at instant t, with heading orientation θ_t.

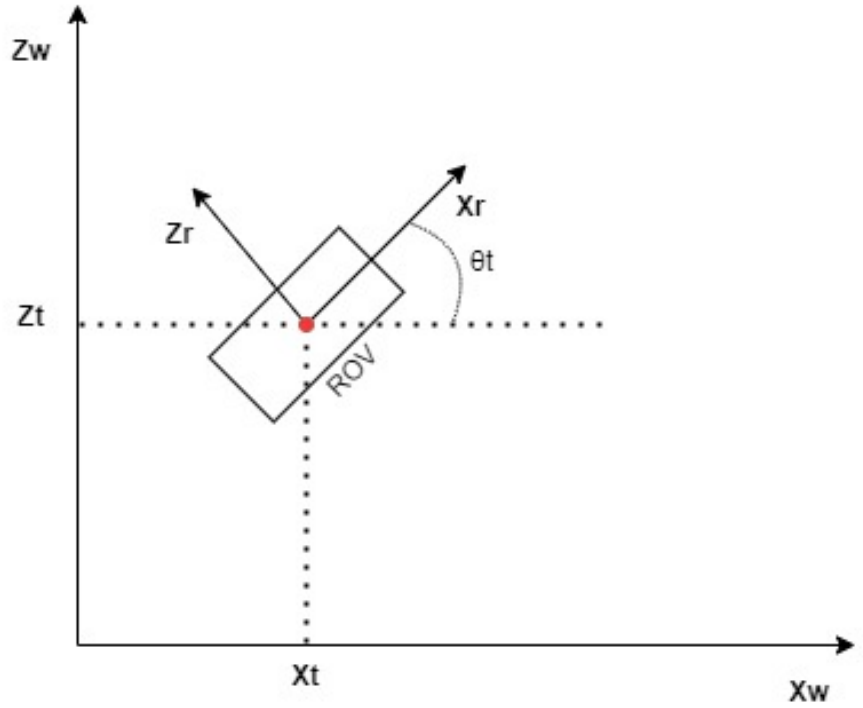

Figure 5.13: Movement of the ROV in the plane with respect to world's coordinates

The following ROV's evolution model was derived:

$$\begin{cases} x_t = x_{t-1} + dx_{r_t} cos(\theta_{t-1} + \omega_t) \\ z_t = z_{t-1} + dz_{r_t} sin(\theta_{t-1} + \omega_t) \\ \theta_t = \theta_{t-1} + \omega_t \end{cases} \tag{5.10}$$

where dx_{r_t} and dz_{r_t} are the ROV's displacement between $t-1$ and t in its coordinates provided by the transitions matrix of the VO. ω_t is the yaw angle of the ROV, in its coordinates and it is recovered by the rotation matrix R with the following equation:

$$\omega = \arctan \frac{R_{21}}{R_{11}} \tag{5.11}$$

The state vector of this system chosen is $X_t = [x_t, z_t, \theta_t]$ and the control input $u_t = (dx_{r_t}, dz_{r_t}, \omega_t)$, provided by VO. State update equation can be rewritten as:

$$X_t = f(X_{t-1}, u_t) + \alpha_t \tag{5.12}$$

with α_t as model noise. Since state update equation is not linear, as explained

in 3.4, the Jacobian of the state update was computed, with respect X_{t-1} and u_t, obtaining the covariance matrix of the motion P_t (see 3.11).
As measurement input were used GPS observations, properly converted for the world's reference system. The H matrix of observations (see 3.9) derived was:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.13}$$

Since GPS has a measurement rate of $1Hz$ and VO inputs have a rate of $1.25Hz$ a synchronization was made. The EKF performs the measurement step only when a GPS input is available, meanwhile it predicts each time a new input arrives from VO. With this criteria, the system is robust when a VO frame is not available (due to some skipped frames in the VO algorithm) or if a GPS input is missing (caused by reception problems).
Experimentally were assigned values of the covariances for process noise and measurement noise. The process noise covariance selected was: $\sigma_{process} = 0.1$, implying a lot of confidence in the model evolution. Covariances of VO inputs were assigned in the following way: $\sigma_{VO_x} = 0.1$, $\sigma_{VO_z} = 0.1$ and $\sigma_{VO_\omega} = 2$ as consequence of VO results (see 5.3.4). Covariance of GPS measurement was assigned as $\sigma_{GPS} = 1$.

5.3.4 Results

In this section the results of the Visual Odometry algorithm alone and its fusion with the GPS, using the EKF are illustrated.

Visual Odometry

The results proposed in this section are focused on the difference between performances of the VO algorithm at various frame rates.
Four different frame rates were tested:

1. Figure 5.14 shows $25Hz$ as update rate for the camera
2. Figure 5.15 shows $5Hz$ as update rate for the camera
3. Figure 5.16 shows $2.5Hz$ as update rate for the camera
4. Figure 5.17 shows $1.25Hz$ as update rate for the camera

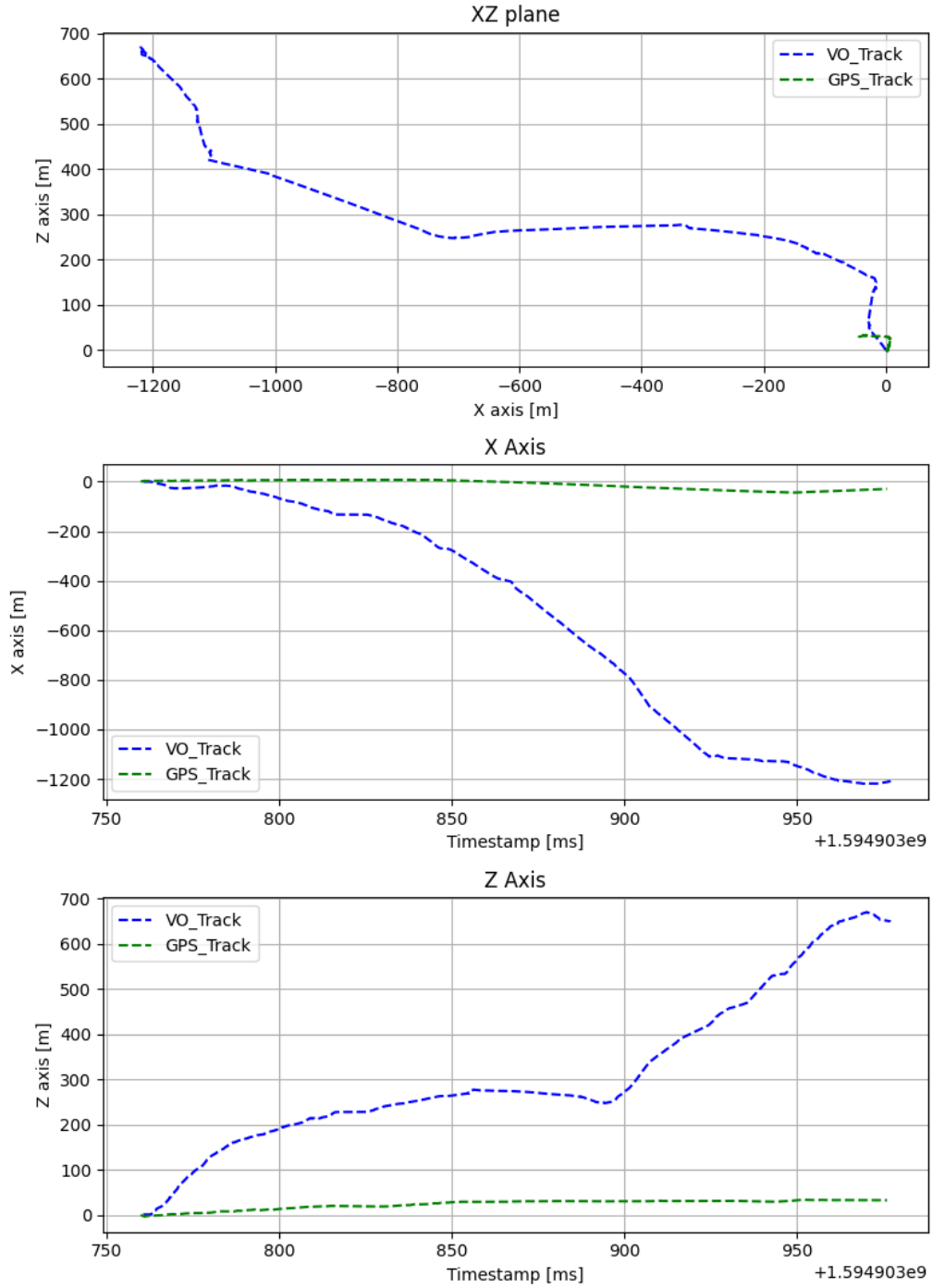

Figure 5.14: VO track compared with GPS ground truth with frame rate fixed to $25Hz$

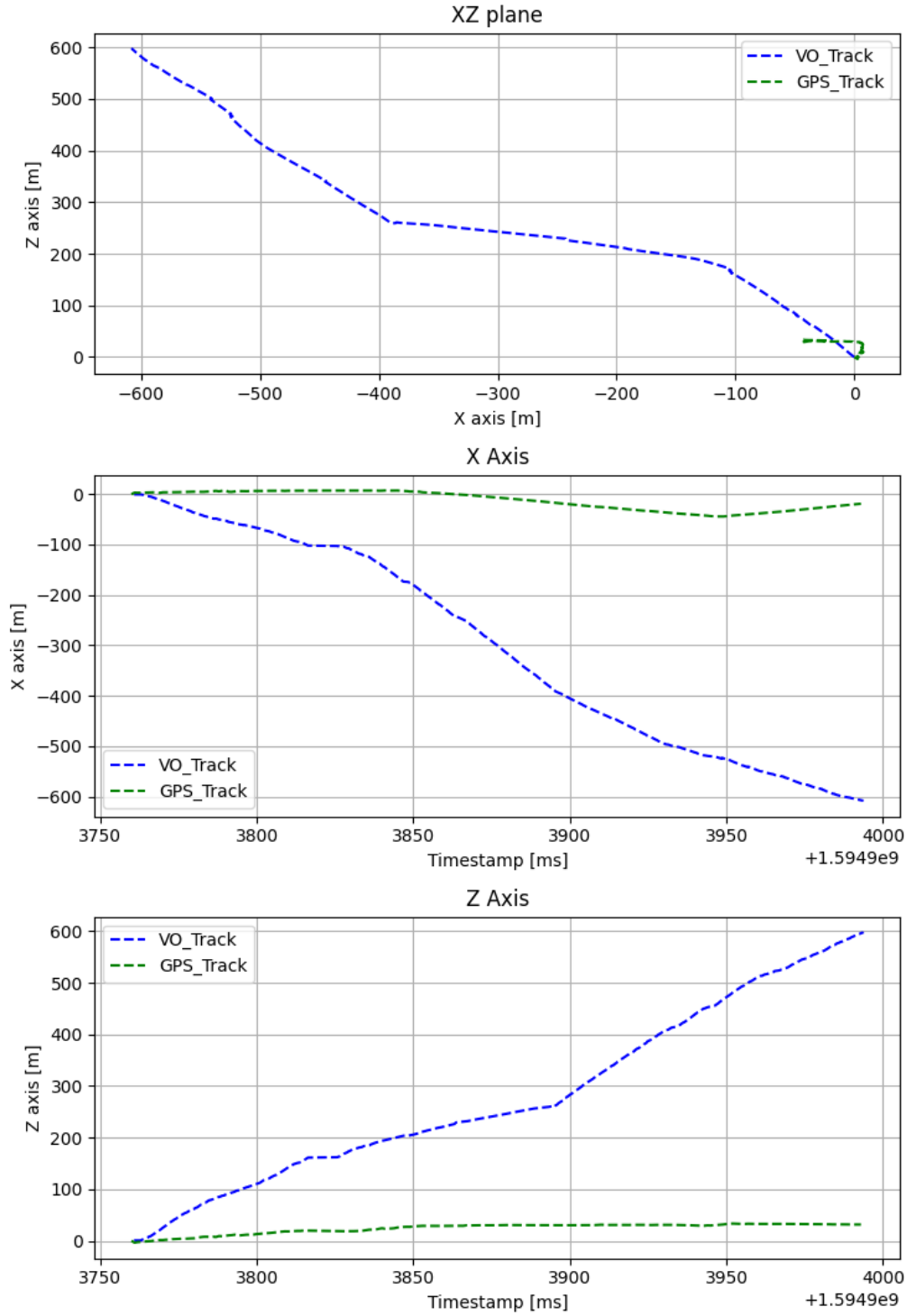

Figure 5.15: VO track compared with GPS ground truth with frame rate fixed to $5Hz$

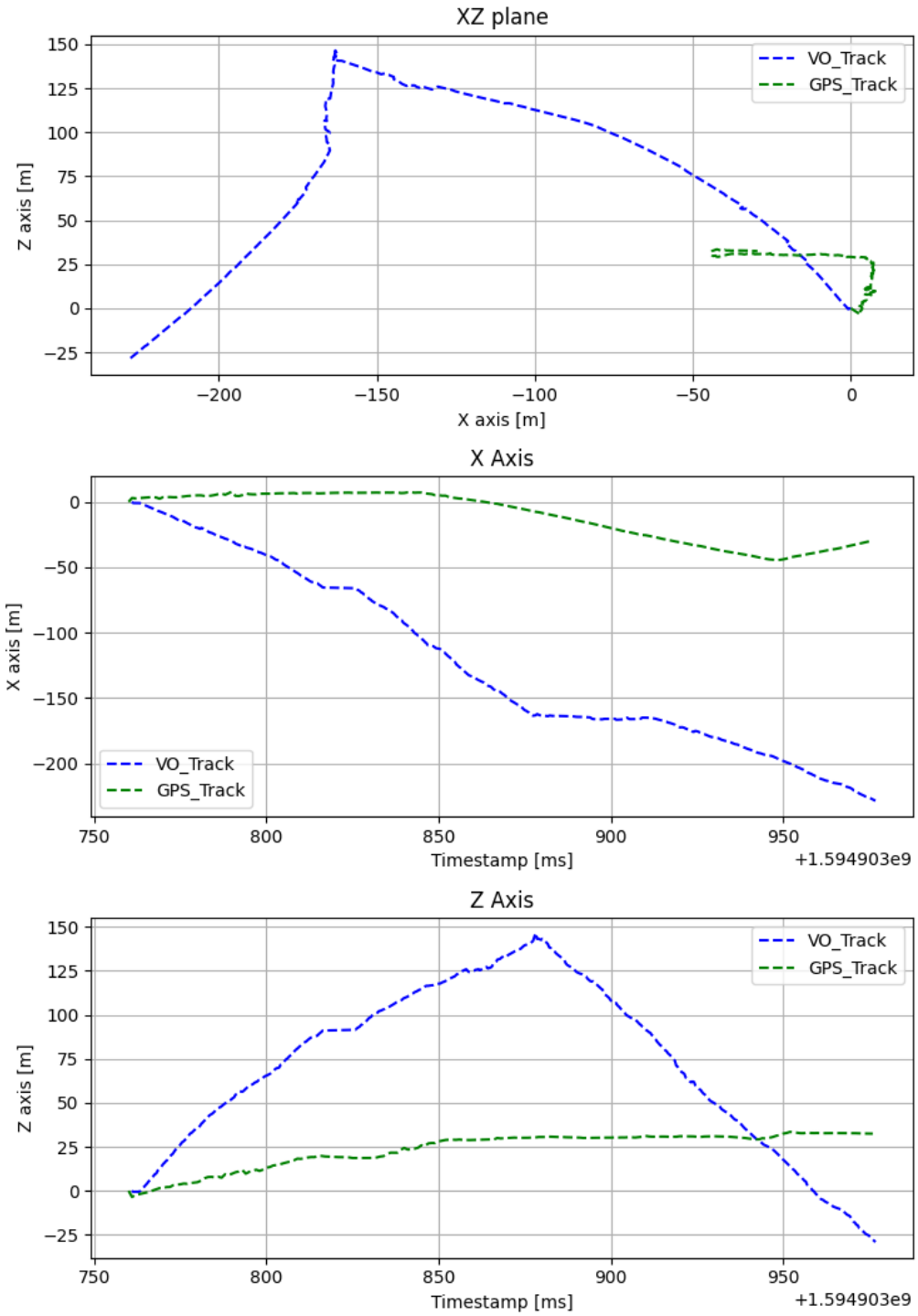

Figure 5.16: VO track compared with GPS ground truth with frame rate fixed to $2.5Hz$

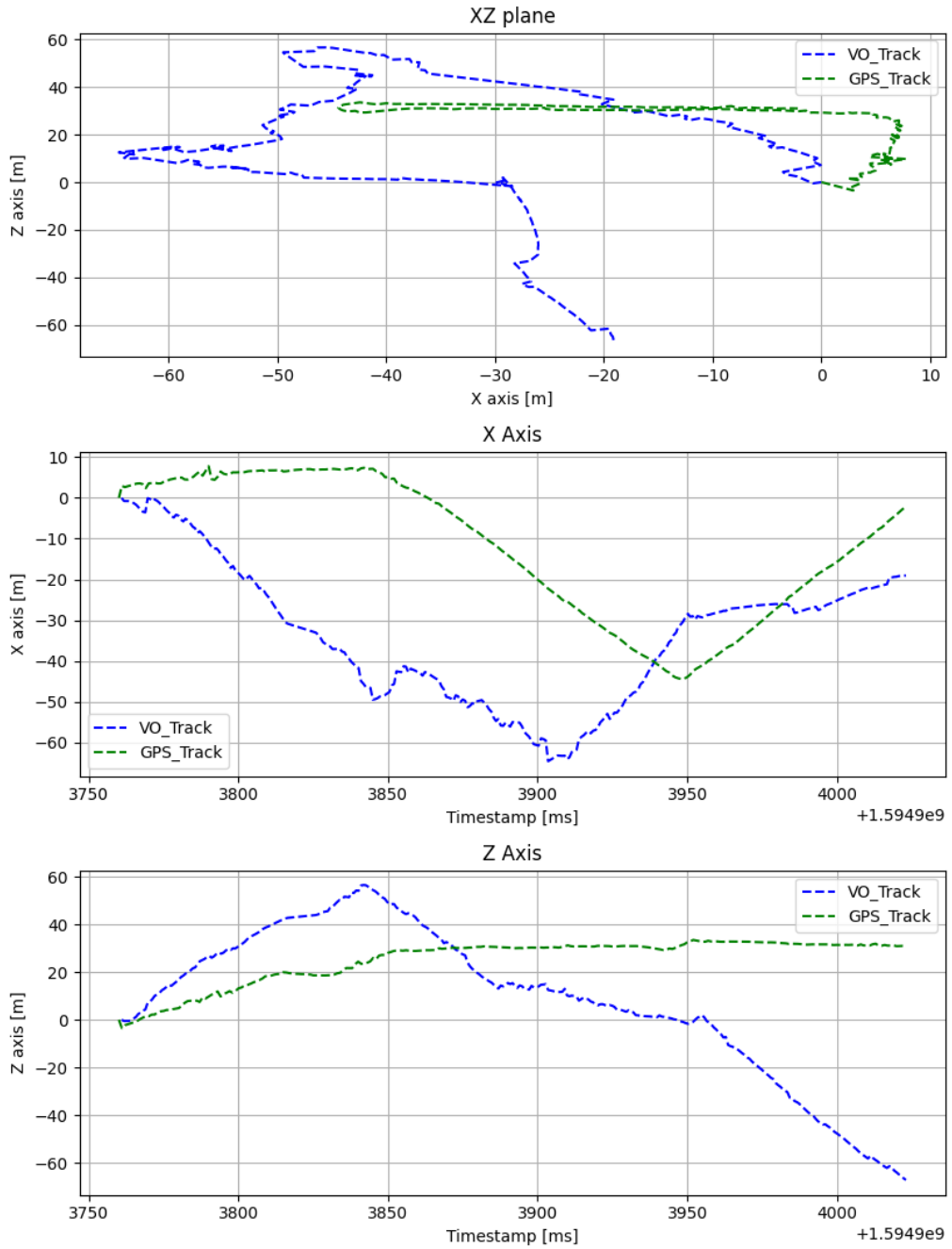

Figure 5.17: VO track compared with GPS ground truth with frame rate fixed to $1.25Hz$

In each graph are represented Visual Odometry's position estimation and GPS ground truth for the XZ plane, and the comparison for the two axes over the time,

in order to have a more detailed indication of the error in the position's estimation. From these results it can be pointed out that the performance of the VO algorithm are inversely proportional to the frame rate. With higher frame rates, the algorithm has a tendency to register important drifts. The consequence of these drifts is that the estimated track derives from ground truth track, with position's estimation that is wrongly guessed up to 10 times with respect GPS track. This is caused, most likely, by the fact that the drifting is correlated with the total number of frames. With the high frame rate there will be a elevate number of consecutive frames to analyze and since VO is a dead reckoning method there is no way to correct these errors. Since the displacement sequence was recorded for approximately 7 minutes, to keep the number of total frames limited, the frame rate was fixed to $1.25Hz$.
Looking at figure 5.17 it is clear that the heading estimation is not correct. In particular in the X axis, at the beginning, heading is almost the opposite of the ground truth but after that error, starts again to estimate correctly the trend. A possible explanation of that is that during one of the turns the movement speed was too fast and the frames captured during this interval had not many common features between them, causing an error in the tracking phase of the features between two consecutive frames.
The results shown that the VO algorithm alone is not too accurate and, for this reason, the integration with the EKF was implemented.

Visual Odometry and GPS fusion

In figure 5.18 are represented the results obtained with the integration of VO inputs with GPS measurements using the EKF described in 5.3.3. The outcome of this test is very promising. It is clear that the position's estimation given by the output of the EKF follows the evolution of the GPS ground truth track. This is caused by the high confidence that the filter has on the measurements, and also by the little confidence assigned to the heading input given by VO. There is a little bit of oscillation of the estimate where input of VO has the opposite direction of the GPS measurement, caused by the correction of the prediction estimate.

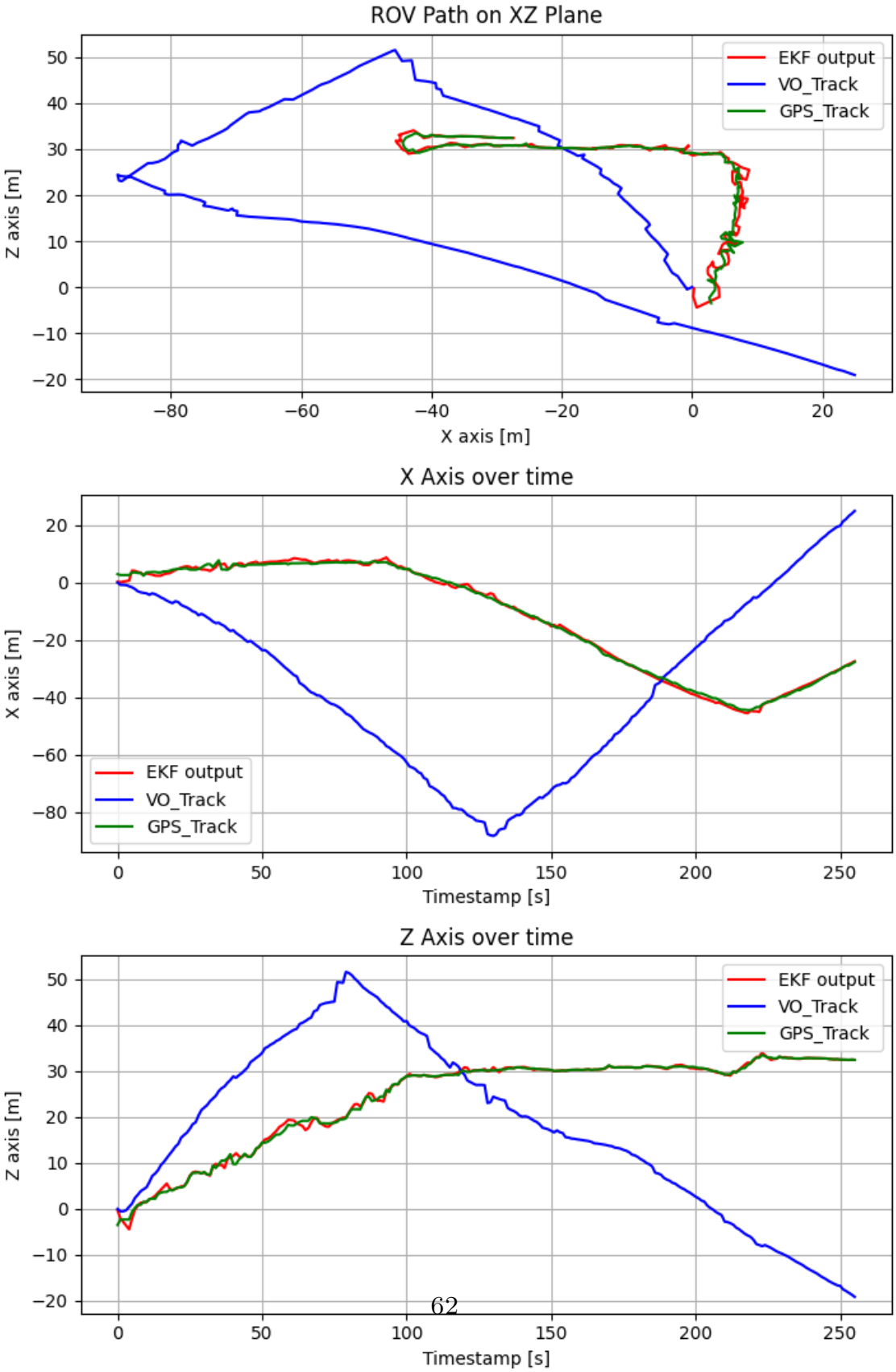

Figure 5.18: Comparison of filtered track, VO track and GPS track

5.4 Underwater test

In this section is presented the underwater test of the visual-inertial algorithm, previously described. In details, is elucidated the construction of the waterproof platform and how the data was acquired. Finally, the results of the test are illustrated.

5.4.1 Waterproof platform

Due to the pandemic situation undergoing during this thesis's project, underwater test's were not possible, since all the public pools were closed, and was not feasible to install a temporary one inside the Polytechnic. A workaround for this problem was found, making the underwater acquisition in a pre-existing drain situated in the hydraulic laboratory, which is typically used for wave tests. The drain has a length of about 40 meters, it can be filled with water for a depth of 1 meter, but has a width of 0.9 meters. That characteristic prevented to use the previously employed BlueROV2. An acquisition platform of reduced dimensions was constructed for this reason. The waterproof enclosure employed was an acrylic tube of 4" provided by BlueRobotics. Inside this tube were placed a Raspberry Pi 4, responsible of the acquisition of the images from the *Raspberry Pi Cam*, and of the IMU's data. The IMU employed was the *MPU 6050*, connected with the Raspberry through I2C protocol. The camera was placed behind the acrylic dome to minimize the distortion effects induced by the difference between water and air refractive indices. The system was powered using an inside battery. In order to place the components in fixed positions, to prevent from misreadings and change of orientations, 3D customs parts were designed and printed, with the help of Team PoliTOcean's [1] mechanical department. Details of the assemblage are shown figure 5.19. The green led that can be seen is the one of the IMU, that is fixed to the bottom.

[1]PoliTOcean is a student team of Polytechnic of Turin that works on underwater robotics

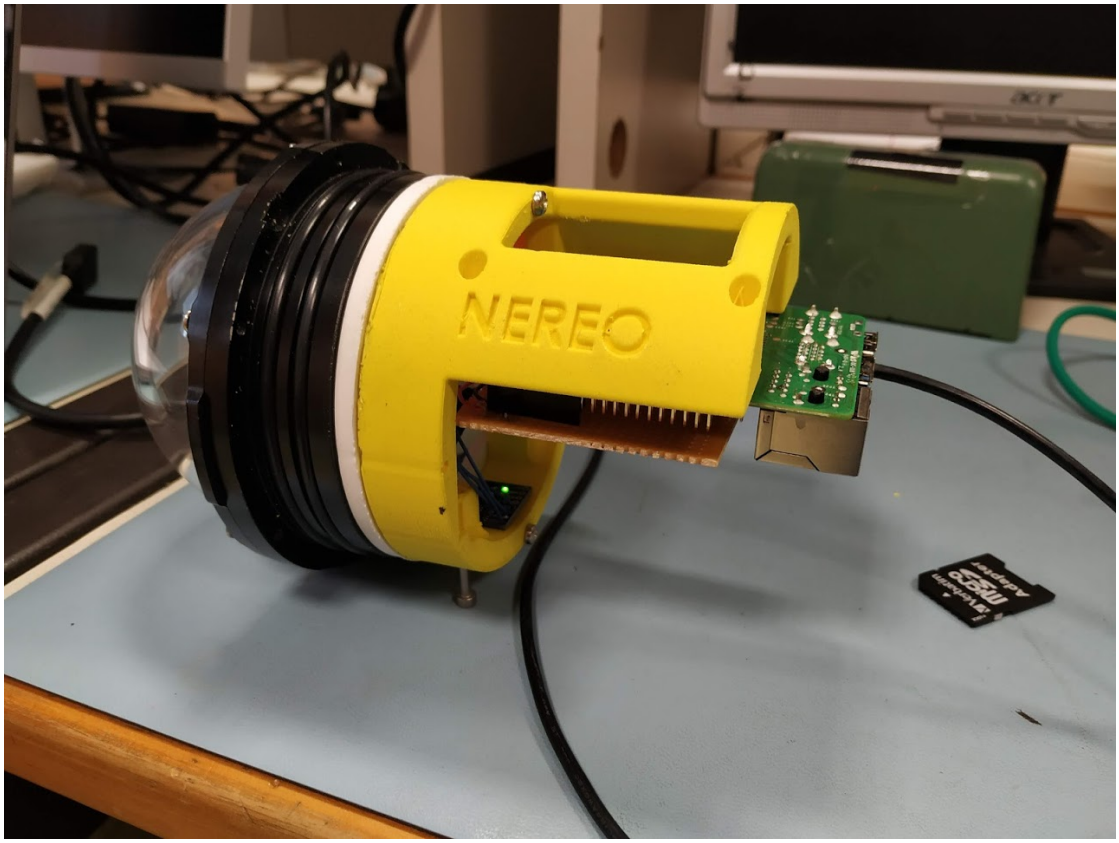

Figure 5.19: Construction phase of the underwater platform

The Raspberry Pi Camera is a low cost camera, that mounts a *Sony IMX219* sensor. Its integration with the Raspberry is easy to do with a dedicated Python library that offers the possibility to change its default parameters on various settings such as: exposure, ISO value, saturation and contrast. This camera was calibrated as illustrated in 4.1, retrieving its intrinsic parameters. To simplify the acquisition part, a Python script was developed: it starts camera acquisition, storing the starting timestamp, whilst it stores every data sent from the IMU with the relative timestamp. In this way the data from the two sensors are synchronized.

5.4.2 Data acquisition

For this test four different datasets were created. The platform was anchored to a cart at fixed height that can be moved by hand, with the help of a rope. The cart slides, taking advantage of the binary that are fixed on the top of the drain, as can be seen in figure 5.20.

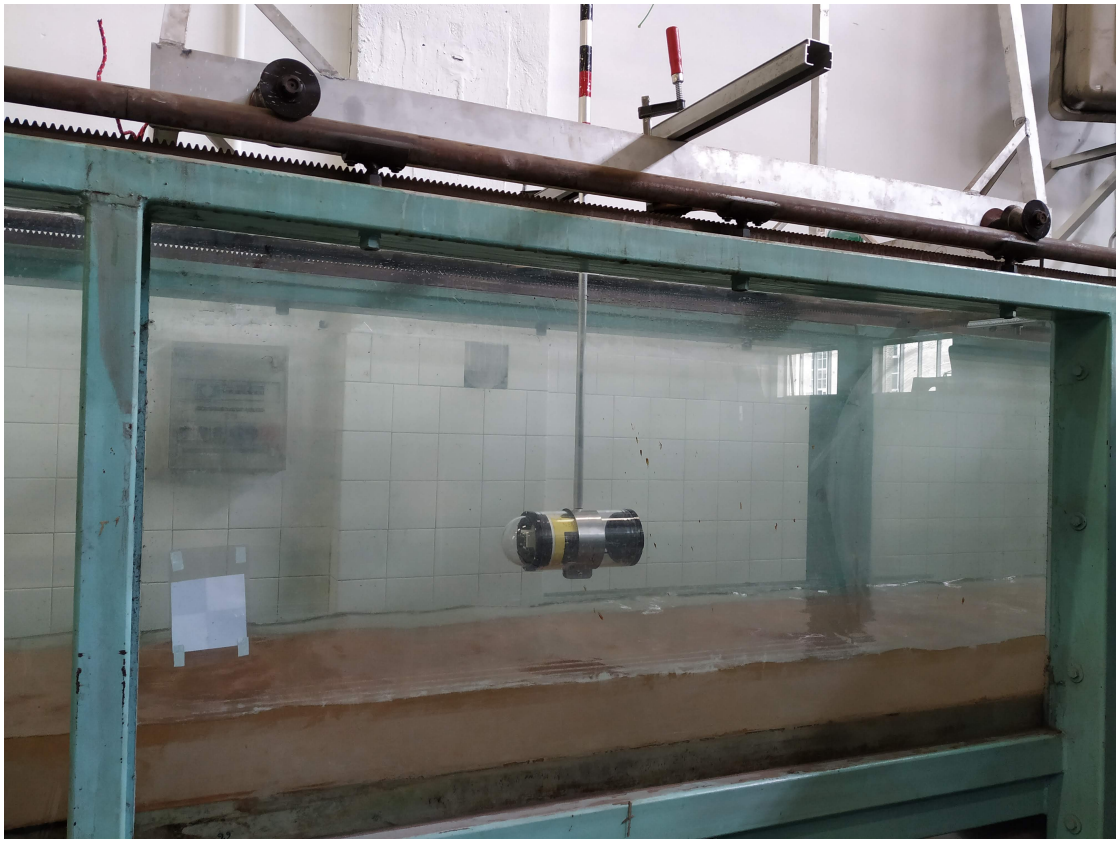

Figure 5.20: Movement system for the platform

With the illustrated architecture the only possible motion of the platform was alongside its X-axis. The objective was to produce a motion that was the most ideal possible, in order to test the effectiveness of the combination of VO and INS with the proposed hardware in an underwater environment.
To the summit of the car, perpendicular to the center of the acquisition platform, a 360 degrees Leica prism was anchored. This prism was needed to track the movement of the platform, in order to have a ground truth track.
A Leica total station [2], placed in a way that had in its field of view the prism, was employed to perform the tracking phase, as can be seen from figure 5.21

[2]Total station is an electronic/optical instrument used for surveying and building construction that can be used to retrieve the coordinates of an unknown point relative to a known coordinate

Figure 5.21: Tracking the movement of the platform with the total station

The four datasets created were repetitions of the same forward movement, having a total lenght each time of about 18 meters. The movement was characterized by a non uniform acceleration and velocity, so the each set has a different duration. Inside of the drain were placed, on the internal sides, several markers that were measured in absolute coordinates with the total station. This choice was made to have additional points that can be used as features for the VO algorithm. Datasets were acquired setting camera frame rate to $25Hz$ and IMU sampling rate to $200Hz$. Camera's settings were kept to default values whilst IMU's LPF was set to $10Hz$.

5.4.3 Results

In figure 5.22 is illustrated the the output of the Kalman filter (in red), derived in 5.1, compared to the ground truth track, the VO alone estimation and the INS navigation. IMU navigation is reconstructed integrating trough the time IMU's readings, after the process of mechanization, in order to simulate the path reconstruction obtained with only the inertial sensor and focus on the contribution of the sensor to the VO-Inertial fusion. It is clear that the output is not quite comparable with respect the ground truth. The cause of this large error in the estimation is attributable to the large derives of the IMU's readings and the wrong estimations of the VO. This results are relative to the first dataset created, but are quite similar to the one obtained with the remaining three.

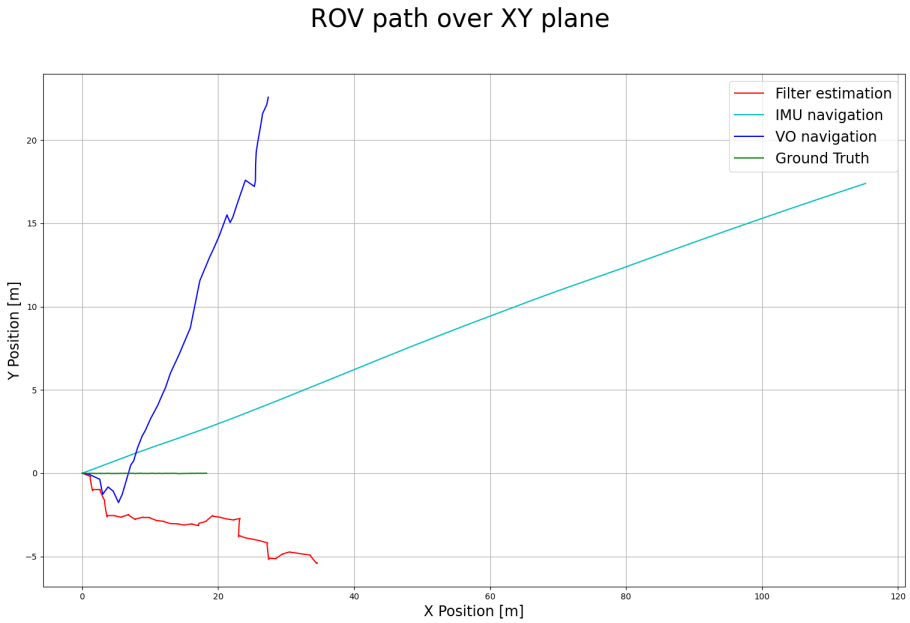

Figure 5.22: Position's estimate: planar path

To understand better the behavior of the algorithm and the cause of the errors the plots for X-axis and Y-axis of the global frame were plotted.
Figure 5.23 shows that the estimation alongside the X-axis is almost similar to the ground truth. The algorithm in this case estimates correctly the position for the first half of the path, then it starts to drift away from the ground truth, changing the slope, particularly from $20s$ to $30s$. This is caused by the combination of the erroneous VO's reconstruction, that worked sufficiently good for the first half, with the fully improper inertial reconstruction.

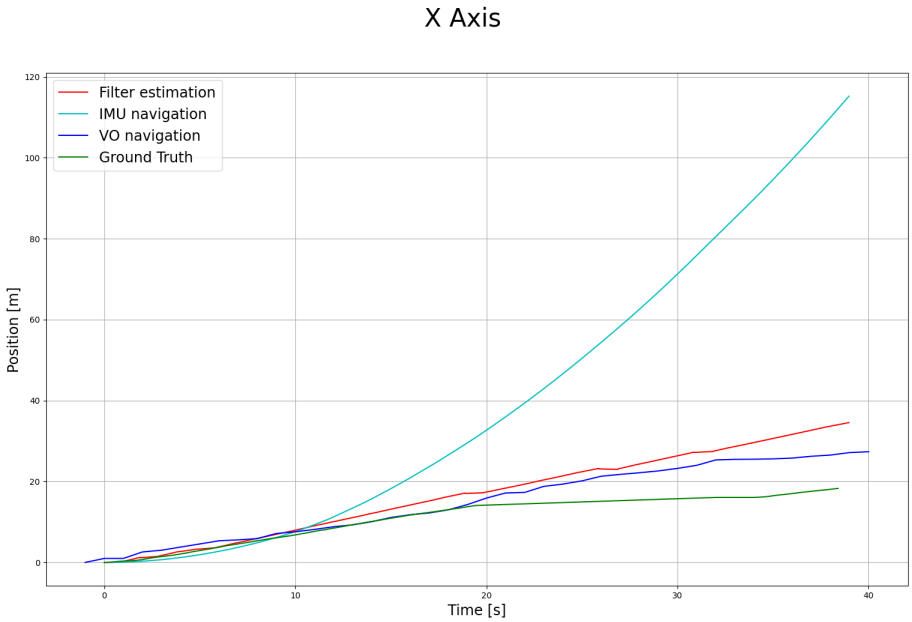

Figure 5.23: Position's estimate: X axis

In figure 5.24 is visible the incorrect estimation of the VO alongside Y-axis that combined with the inertial navigation gives a non acceptable result.

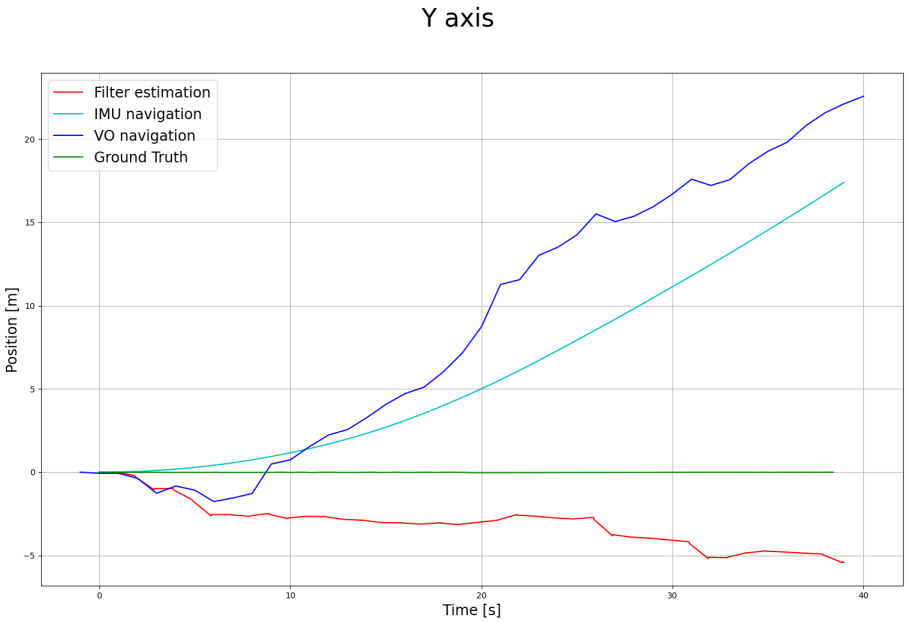

Figure 5.24: Position's estimate: Y axis

Figure 5.25 is remarkably useful to understand better the causes of this result. From X-accelerometer and velocity graphs it can be noticed that the readings are mostly noisy, with a important oscillation when the platform is moving. This oscillations cause, when they are integrated, a completely wrong evaluation of the velocity. In the tested scenario, even a small error on the measure of the acceleration can cause a disastrous effect. On the other hand, VO's velocities reading are similar to the ground truth ones, and at each measurement step of the filter correct the prediction done with IMU's reading.
Similar discussion can be made for the Y-axis: the acceleration's readings suffers from biases and noises that affects the velocity estimation. Even in this case, VO's readings are used to correct the prediction.

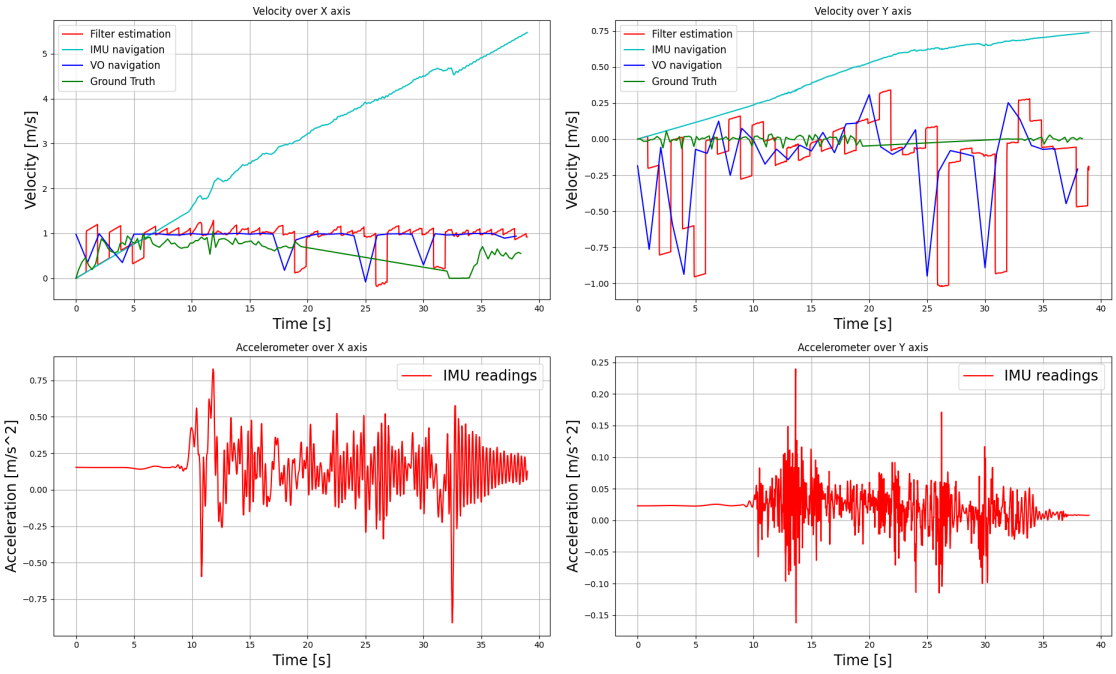

Figure 5.25: Velocity results of the algorithm and inputs of the accelerometer

Poor performances of this system can be credited to untrustworthy IMU's outputs. This inertial sensor is not indicated for small variations of acceleration, such as the one registered in this test.
Visual Odometry performance, on the other side, is not the best in this underwater environment. This is caused by the low image quality, that leads to a reduced number of features that can be found. In figure 5.26 is visible that the mean value of features found over the video recording is reduced compared to the one obtained with the Aqualoc dataset in 5.2.3. It can be noticed that in some cases the ORB matcher fails to find similar features between consecutive frames, and even when it accomplish that, the overall number of matches is limited.

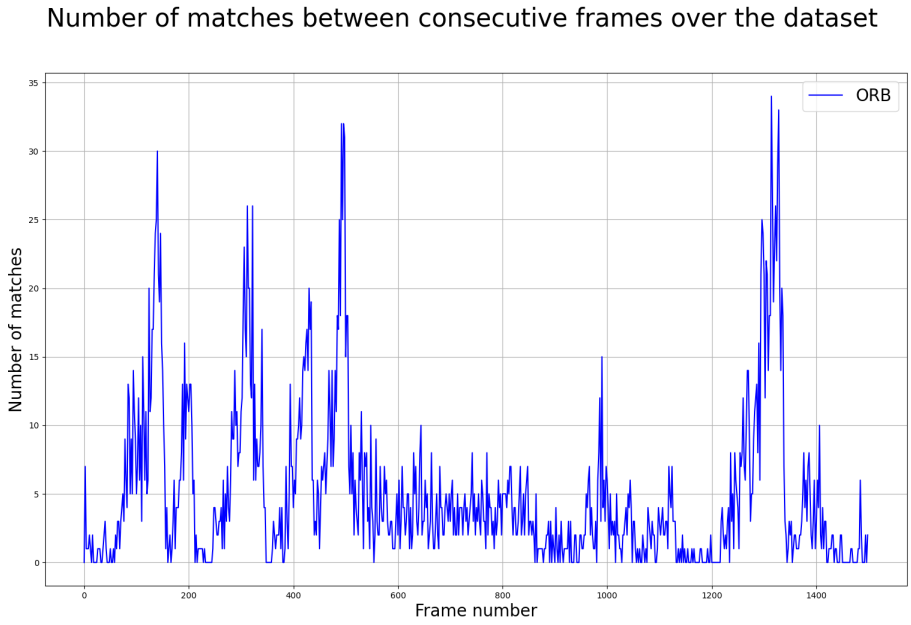

Figure 5.26: Performances of ORB over the dataset acquired

Chapter 6

Conclusions

UUVs are nowadays more and more used to perform visual inspections, maintenance and repair of dams, pipes, tunnels, structures as well as the analyses of the underwater environments in lakes,rivers and seas.
Underwater positioning represents a huge challenge, since the impossibility to exploit common radio-based techniques, used in terrestrial environments, such as GPS. The reason of this problem is that the radio signals are jammed from the water.
Alternative positioning systems are used to overcome this obstacle. They are typically based on active acoustic signals. These solutions are expensive and the sensor's size specifications prevent their integration within small mobile systems, requiring also a high computational power.
In this study was presented a low-cost positioning system for an underwater environment, using a combination of Inertial Navigation and Visual Odometry based navigation. The sensor's fusion was done with an Extended Kalman Filter. The hardware employed consisted in a Raspberry Pi 4, which is a common andaffordable embedded computer, a low-cost digital camera and an entry level IMU.
The initial part of this study, focused on sensor's characterization showed that IMU's measurements were affected by drifts and noises. The inputs were denoised, and the offset observed in stationary conditions was removed, in order to have the highest precision possible of IMU's readings.
Feature detectors, from the test done on the AQUALOC dataset (5.2.3), performed well in the underwater scenario, especially after the image enhancement done on each frame, showing an good average match ratio over the dataset.
The work done on VO performance, evaluated in the terrestrial scenario, revealed good results, specially with the pairing of the VO output with the GPS measurements, reaching an high accuracy of path estimation (5.3.4).
Unfortunately the results obtained working in the underwater environment (5.4) proved that the combination of camera and inertial sensor is not suitable for this

application, with the selected hardware. Indeed the path's reconstruction is not as good as expected, since the estimation over the Y-axis derives from ground truth by almost 5 meters, as shown in figure 5.24. Therefore the trajectory's estimation is not reliable over the underwater navigation. The Extended Kalman Filter works well, since the dynamic motion in its prediction phase, filters as high as possible the noise from inertial's readings and VO's estimations, as can be seen in the estimation of X-axis path in figure 5.23.
Explanation of this performance can be found in poor camera's image quality (as shown in figure 5.26) and in the noises plus drifts of IMU's measurements. IMU's noises cause, when they are integrated, a completely wrong evaluation of the velocity estimation that itself leads to an erroneous position estimation. In the tested scenario, even a small error on the measure of the acceleration can cause a disastrous effect, since the slow dynamic of the ROV and the short overall length of the movement.
To overcome this problem a suitable solution might be to replace the camera employed with a more powerful one. For example a stereo camera would ideally improve quality of VO estimations, since it can retrieve object's scale accurately. This will increase the cost of the hardware, since this kind of camera is quite expensive and its integration requires an high-performance on board computer, the low cost one used would not be sufficient to accomplish this task. As discussed in [41], stereo VO's performances in an underwater environment are promising, showing a good accuracy of pose estimation. Furthermore, it might be useful to replace the IMU with a model that has more accuracy of its measurements. As well as replacing the camera, this operation will increase the cost of the hardware, considering that high precision IMUs cost up to 100 times the value of the one employed in this project.
The above presented solutions are beyond the purpose of this master's thesis work, since the challenge was to find an alternative of underwater positioning using affordable components.
Moreover, an alternative solution might be the addition of more low-cost sensors to the ones used, for example a depth sensor and a compass sensor can give an improvement of the estimation of the position. The addition of the depth sensor can give an extra input in the Kalman Filter, providing the measurement of the displacement alongside the Z-axis. The compass sensor, instead, would be useful to correct the heading estimation of the Kalman Filter, since it returns the orientation of the ROV. However the challenging part would be to keep a low computational time using the same embedded computer, in order to work in a nearly real-time application, since more measurements steps will increase the computational load of the algorithm.

Bibliography

[1] URL: https://oceanservice.noaa.gov/facts/exploration.html (cit. on p. 3).

[2] URL: https://www.offshore-mag.com/subsea/article/14179171/oceaneering-to-supply-m5-connectors-for-offshore-western-australia-project (cit. on p. 4).

[3] URL: https://www.oceaneering.com/rov-services/rov-systems/ (cit. on p. 5).

[4] URL: https://www.lerus-training.com/blog/wp-content/uploads/2015/07/Class-4-ROV-.jpg (cit. on p. 5).

[5] URL: https://www.hydroid.com/news/hydroid-introduces-new-generation-remus-100-auv (cit. on p. 6).

[6] Jianhua Bao, Daoliang Li, Xi Qiao, and Thomas Rauschenbach. «Integrated navigation for autonomous underwater vehicles in aquaculture: A review». In: *Information Processing in Agriculture* 7.1 (2020), pp. 139–151. ISSN: 2214-3173. DOI: https://doi.org/10.1016/j.inpa.2019.04.003. URL: http://www.sciencedirect.com/science/article/pii/S221431731930071X (cit. on p. 7).

[7] URL: https://www.nortekgroup.com/knowledge-center/wiki/new-to-subsea-navigation (cit. on p. 7).

[8] B. Benson, Y. Li, R. Kastner, B. Faunce, K. Domond, D. Kimball, and C. Schurgers. «Design of a low-cost, underwater acoustic modem for short-range sensor networks». In: *OCEANS'10 IEEE SYDNEY*. 2010, pp. 1–9. DOI: 10.1109/OCEANSSYD.2010.5603816 (cit. on p. 7).

[9] Qiang Bai Yong Bai. *Subsea Engineering Handbook*. Gulf Professional Publishing, 2012. Chap. 4 (cit. on p. 8).

[10] URL: https://bluerobotics.com/learn/understanding-and-using-scanning-sonars/ (cit. on p. 8).

[11] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS - Global Navigation Satellite Systems : GPS, GLONASS, Galileo and more*. japanisch. Translation of the first edition of the Springer Publishing Com. Deutschland: Springer Verlag, 2008 (cit. on p. 9).

[12] B. Alsadik. «Multibeam Lidar for mobile mapping systems». English. In: *GIM International* (Aug. 2020), pp. 1–3. ISSN: 1566-9076 (cit. on p. 9).

[13] Kenneth Gade. «Inertial Navigation — Theory and Applications». PhD thesis. Jan. 2018 (cit. on p. 9).

[14] Alberto Quattrini Li et al. «Experimental Comparison of Open Source Vision-Based State Estimation Algorithms». In: Mar. 2017, pp. 775–786. ISBN: 978-3-319-50114-7. DOI: 10.1007/978-3-319-50115-4_67 (cit. on p. 10).

[15] E. Salahat and M. Qasaimeh. «Recent advances in features extraction and description algorithms: A comprehensive survey». In: *2017 IEEE International Conference on Industrial Technology (ICIT)*. 2017, pp. 1059–1063. DOI: 10.1109/ICIT.2017.7915508 (cit. on p. 11).

[16] Yali Li, Shengjin Wang, Qi Tian, and Xiaoqing Ding. «A survey of recent advances in visual feature detection». In: *Neurocomputing* 149 (2015), pp. 736–751. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2014.08.003. URL: https://www.sciencedirect.com/science/article/pii/S0925231214010121 (cit. on p. 12).

[17] M. Hassaballah, Abdelmgeid Ali, and Hammam Alshazly. «Image Features Detection, Description and Matching». In: vol. 630. Feb. 2016, pp. 11–45. ISBN: ISBN 978-3-319-28852-9. DOI: 10.1007/978-3-319-28854-3_2 (cit. on pp. 12, 13).

[18] Jianbo Shi and Tomasi. «Good features to track». In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794 (cit. on p. 12).

[19] Tony Lindeberg. «Feature Detection with Automatic Scale Selection». In: *International Journal of Computer Vision* 30 (Sept. 1998), pp. 77–116. DOI: 10.1023/A:1008045108935 (cit. on p. 13).

[20] David G. Lowe. «Distinctive Image Features from Scale-Invariant Keypoints». In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94 (cit. on pp. 13, 14).

[21] URL: https://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O (cit. on p. 15).

[22] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. «SURF: Speeded Up Robust Features». In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8 (cit. on p. 15).

[23] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. «BRIEF: Binary Robust Independent Elementary Features». In: vol. 6314. Sept. 2010, pp. 778–792. ISBN: 978-3-642-15560-4. DOI: 10.1007/978-3-642-15561-1_56 (cit. on p. 16).

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. «ORB: An efficient alternative to SIFT or SURF». In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544 (cit. on p. 16).

[25] Jiawang Bian, wen-yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. «GMS: Grid-Based Motion Statistics for Fast, Ultra-Robust Feature Correspondence». In: July 2017, pp. 2828–2837. DOI: 10.1109/CVPR.2017.302 (cit. on p. 17).

[26] C. Tomasi and Takeo Kanade. *Shape and Motion from Image Streams: a Factorization Method – Part 3 Detection and Tracking of Point Features*. Tech. rep. CMU-CS-91-132. Pittsburgh, PA: Carnegie Mellon University, Apr. 1991 (cit. on p. 17).

[27] Martin A. Fischler and Robert C. Bolles. «Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography». In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692 (cit. on p. 18).

[28] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. «An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics». In: *Intelligent Industrial Systems* 1 (Nov. 2015). DOI: 10.1007/s40903-015-0032-7 (cit. on pp. 18–20, 24).

[29] Youngjoo Kim and Hyochoong Bang. «Introduction to Kalman Filter and Its Applications». In: *Introduction and Implementations of the Kalman Filter*. 2019. DOI: 10.5772/intechopen.80600. URL: https://app.dimensions.ai/details/publication/pub.1112011530%20and%20https://www.intechopen.com/citation-pdf-url/63164 (cit. on p. 22).

[30] Xi-Chao Yin, Pu Han, Jun Zhang, Feng-Qi Zhang, and Ning-Ling Wang. «Application of wavelet transform in signal denoising». In: *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*. Vol. 1. 2003, 436–441 Vol.1. DOI: 10.1109/ICMLC.2003.1264517 (cit. on p. 28).

[31] Guomin Luo and Daming Zhang. «Wavelet Denoising». In: Apr. 2012. ISBN: 978-953-51-0494-0. DOI: 10.5772/37424 (cit. on p. 28).

[32] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. «Analysis and Modeling of Inertial Sensors Using Allan Variance». In: *Instrumentation and Measurement, IEEE Transactions on* 57 (Feb. 2008), pp. 140–149. DOI: 10.1109/TIM.2007.908635 (cit. on p. 32).

[33] «IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros». In: *IEEE Std 952-1997* (1998), pp. 1–84. DOI: 10.1109/IEEESTD.1998.86153 (cit. on p. 32).

[34] Andrea Civita, Simone Fiori, and Giuseppe Romani. «A Mobile Acquisition System and a Method for Hips Sway Fluency Assessment». In: *Information* 9 (Dec. 2018), p. 321. DOI: 10.3390/info9120321 (cit. on p. 37).

[35] Scott Gleason, Demoz Gebre-Egziabher, and Demoz Gebre Egziabher. *GNSS Applications and Methods*. English. Artech House, July 2009. ISBN: 1596933293 (cit. on p. 40).

[36] 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* 38.14 (2019), pp. 1549–1559. DOI: 10.1177/0278364919883346. eprint: https://doi.org/10.1177/0278364919883346. URL: https://doi.org/10.1177/0278364919883346 (cit. on p. 42).

[37] URL: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ (cit. on p. 42).

[38] URL: https://opencv.org/about/ (cit. on p. 43).

[39] Y. Zhao, N. D. Georganas, and E. M. Petriu. «Applying Contrast-limited Adaptive Histogram Equalization and integral projection for facial feature enhancement and detection». In: *2010 IEEE Instrumentation Measurement Technology Conference Proceedings*. 2010, pp. 861–866. DOI: 10.1109/IMTC.2010.5488048 (cit. on p. 43).

[40] URL: https://bluerobotics.com/wp-content/uploads/2020/02/br_bluerov2_datasheet_rev6.pdf (cit. on p. 50).

[41] Jun Zhang, Viorela Ila, and Laurent Kneip. «Robust Visual Odometry in Underwater Environment». In: May 2018, pp. 1–9. DOI: 10.1109/OCEANSKOBE.2018.8559452 (cit. on p. 73).