FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Robust Visual Odometry System For Autonomous Surface Vehicles Using Deep Learning

Eduardo Jorge Pereira Gonçalves

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Andry Maykol Gomes Pinto

October 25, 2023

Resumo

A posição geoestratégica portuguesa cria as condições ideais para o teste e desenvolvimento de tecnologia com aplicação direta em ambientes marítimos, deixando-nos a nós, portugueses, na linha da frente da investigação de sistemas robóticos que operem no meio aquático. Esses sistemas têm a capacidade para libertar o Homem de atividades potencialmente perigosas. Os Veículos Autónomos de Superfície (ASVs), capazes de monitorizar, operar e navegar de forma autónoma, dependem, no entanto, de uma estimação precisa da localização.

O presente trabalho foca-se na área da estimação da localização de ASVs. Dadas as restrições impostas por ambientes marítimos, é sugerida uma abordagem baseada em fusão sensorial, por forma a criar um sistema de localização mais robusto com informação multi-modal, nomeadamente informação GPS/INS, LiDAR e visual. Para isso, são utilizadas técnicas Deep Learning, com destaque para mecanismos de atenção e codificação posicional, para fundir informação com tendência a falhas, temporalmente desalinhada e irregularmente espaçada.

A alteração da codificação puramente posicional para uma codificação posicional baseada na informação temporal associada às estimativas, dimiuiu o erro de treino dos modelos em cerca de 50%. Os sistemas propostos melhoraram as estimativas de localização, num dataset recolhido em ambiente real, em 7.7% e 20.7% em translação e rotação, respetivamente, em comparação com o melhor dos métodos utilizados como input do sistema de fusão (odometria baseada em informação LiDAR).

Os resultados obtidos comprovam a eficácia do modelo em lidar com dados recolhidos em ambientes altamente dinâmicos, com tendências a falhas e defeitos, próprios de cenários portuários, aumentando a precisão da estimação da localização de ASVs.

Abstract

The Portuguese's geostrategic position creates ideal conditions to the testing and the development of maritime-based technology, leaving us, the Portuguese, on the front line of research in the area of maritime robotics. These systems can release humans from potentially dangerous activities. Autonomous Surface Vehicles, ASVs, are systems that are able to monitor, operate and navigate in an autonomous fashion. These systems, however, are highly dependent on a precise localisation estimate.

The present work focuses on the field of the localisation of ASVs. Given the harsh conditions created by maritime environments, an approach based on sensor fusion techniques is proposed, so that a robust multimodal information localisation estimation system is created, namely with GPS/INS, LiDAR, and visual information. For that, Deep Learning techniques are used, with highlight to attention mechanisms and position encoding, so that information that is prone to failures, temporally misaligned and irregularly spaced can be fused.

The change of a purely positional encoding to a temporally-based positional information decreased training error by, approximately, 50%. The proposed systems improved localisation estimates, in a dataset collected in a real-world environment, by 7.7% and 20.7% in translation and rotation, respectively, when compared to the best localisation estimation method used as input to the fusion system (LiDAR-based odometry).

The results prove the fusion system's ability to deal with data collected in highly dynamic environments, prone to failures and defects, typically seen in data collected in harbor scenarios, increasing the precision of the localisation estimation tasks of ASVs.

Agradecimentos

Ao meu orientador, Professor Andry Pinto, pela disponibilidade, acompanhamento e, acima de tudo, pela motivação e confiança demonstrada, mesmo nos momentos mais difícies deste percurso. A toda a equipa do CRAS que comigo lidou, em especial à Maria Inês Pereira e ao Daniel Campos, que tiveram sempre tempo e conselhos para me dar.

À minha mãe, que tanto tem sacrificado em prol do bem estar dos filhos, obrigado pela paciência e compreensão. Ao meu irmão pela companhia e aos meus avós pelo amor e carinho. Foi o vosso apoio diário e incondicional que não me deixou desistir.

A todos os familiares e amigos, mais ou menos próximos, que me incentivaram e aconselharam, um enorme obrigado.

Eduardo Gonçalves

Official Acknowledgments

This thesis was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 871571.

Eduardo Gonçalves

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
ASV	Autonomous Surface Vehicle
CNN	Convolutional Neural Network
DL	Deep Learning
DoF	Degrees Of Freedom
EKF	Extended Kalman Filter
EMSA	European Maritime Safety Agency
FAST	Features from Accelerated Segment Test
FoV	Field of View
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LiDAR	Light Detection And Ranging
LSTM	Long Short-Term Memory
MAP	Maximum a Posteriori
MDN	Mixture Density Network
MLP	Multilayer Perceptron
NLP	Natural Language Processing
O&M	Operations and Maintenance
ORB	Oriented FAST and rotated BRIEF
Radar	Radio Detection and Ranging
RCNN	Recurrent Convolutional Neural Network
RL	Robot Localization
RMS	Root Mean Square
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROS	Robot Operating System
RTK	Real Time Kinematic
SLAM	Simultaneous Localization And Mapping
SLERP	Spherical Linear Interpolation
Sonar	Sound Navigation and Ranging
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle
VO	Visual Odometry
V-SLAM	Visual Simultaneous Localization And Mapping

Chapter 1

Introduction

1.1 Context

For centuries, Portugal has had a close relationship with the sea. Close to half of its borders are surrounded by the Atlantic Ocean, and its autonomous regions - Madeira and Azores - make it possible for Portugal to have one of the biggest Exclusive Economic Zones in Europe[1].Not only does it constitute a substantial economic resource, but it also represents a tremendous responsibility. The extensive area has to be monitored, studied and, eventually, protected. Additionally, the ever-increasing needs of the global market have led to a rising number of merchant ships sailing in deep waters. These factors translate into a large number of human resources and, therefore, into high operational costs, which could be reduced if the vehicles were autonomous [1]. The benefits of reducing the number of crew members and, for that reason, the number of human-made decisions, go well beyond the economic factor. Indeed, EMSA estimated that human action was the main cause of marine incidents occurred between 2014 and 2020, with system or equipment failure coming second on the list [2].

Uncrewed ships or vessels could also reduce the number of casualties in case of an incident, releasing humans from potentially dangerous functions. Aiming for a future where risks and costs associated with sea exploration are reduced, Autonomous Surface Vehicles (ASVs) come up as an appealing solution. An ASV (fig. 1.1) is an autonomous vehicle that operates on the surface of the water. Usually performing advanced tasks in a completely autonomous fashion, ASVs have been used for numerous purposes. The scientific community, for instance, has been developing and using such vehicles for several applications, such as seafloor mapping operations, pollutant detection, structure inspection [2, 3], harbor monitoring operations, structure recognition [4], etc. Such vehicles are also common in the military, carrying patrolling tasks for border and coastal control.

[1] Resolução do Conselho de Ministros n.º 68/2021, D.R. I Série. 108 (Accessed Jun. 8, 2023)
[2] Annual overview of marine casualties and inci- dents (Accessed Apr. 20, 2023)

Figure 1.1: SENSE ASV [5]

ASVs rely on an array of sensors that acquire complex information from their surroundings - exteroceptive sensors - and sensors that monitor internal conditions of the vehicle - proprioceptive sensors. The former normally consists of GNSS, Radar, Sonar, camera, and LiDAR systems, while the latter are usually composed of IMUs and motor encoders. Those provide raw data that has to be processed and then used for tasks like path planning [6], task scheduling, obstacle avoidance [7], structure detection [8], and localisation estimation [9]. The scope of this work of this dissertation focuses on the latter, especially on the integration of data from different sensors in a robust Visual Odometry (VO) system. An agent's localisation estimation can face many challenges like it is the case of poor GNSS signals. That is where alternative odometry systems come into hand. VO is the process of estimating the movement of visual cameras based on visual information. The idea first saw light in the decade of 1980 and has since been the target of many different approaches and implementations, given the interest of the scientific community in the fields of computer vision and autonomous systems. The use of such techniques is still not commonplace in maritime navigation, and the examples of application of VO in ASVs are scarce, given the lack of fixed references that allow visual-based algorithms to perceive changes in orientation and position.

Autonomous systems' ability to perform sequences of complex actions is hugely dependent on the quality of the localisation estimation that those systems are able to obtain. A defective estimation can lead to catastrophic results, especially when a system operates subject to tight margins. ASVs are no exception to the necessity of a good position and orientation estimation. The state-of-the-art methods for pose calculation of such vehicles usually rely on GNSS and inertial measurement systems, which can lead to errors superior to a meter in some conditions [10]. Despite being sufficient for the majority of the length of most missions, some situations are constrained to tighter position measurements, as is the case of docking maneuvers. In fact, as a vessel approaches a

harbor, the quality of GNSS measurements tends to degrade due to reflections and attenuation of the signals that are common near architectural structures, precisely in situations where accurate localisation measurements are essential.

Even though architectural structures are obstacles to GNSS measurements, they also represent an opportunity to delve into different sensors with localisation purposes. The use of VO or SLAM implementations is made possible by the characteristics of such environments, thanks to the use of visual cameras and LiDARs as data acquisition systems. However, the application of such light-dependent sensors is not as straightforward as it might appear. Cameras can suffer from distortion issues when fast movements occur, possibly caused by waves and undulation, or suffer from sudden exposure fluctuation, while LiDARs' performance can be severely affected under marine conditions and its typical phenomena, such as in the case of fog formation [11].

1.2 Motivation

Exploring the strengths and debilities of each sensor could increase the quality of the predictions made by a localisation estimation algorithm. On top of that, using different sensors in a complementary fashion could also leverage the robustness of a system, in case of sensor failures. In fact, sensor fusion tries to mimic the way humans perceive the environment based on multi-modal information. Distinct sensing receptors are used simultaneously during activities that require situational awareness capabilities. Indeed, different modalities cooperate with each other, so that humans are able to understand their surroundings in a robust manner. In a similar way, autonomous systems benefit from the fusion of multi-modal information, acquired by heterogeneous sensors. Such heterogeneous sensors produce different types of data that might have to be fused, representing a challenging task, but improving autonomous navigation performance. As a matter of fact, some sensor fusing techniques are already performed, despite some obstacles such implementations may face. Those might be related to data imperfections (e.g. uncertainty in measurements), outliers due to environmental conditions, data alignment (e.g. different sensor frames), different operational frequencies, etc. [12]. Sensor Fusion techniques tackle obstacles like those mentioned above.

For that matter, one of the numerous applications of Deep Learning (DL) techniques focuses on information fusion. Similarly to classical implementations, neural networks can use multi-modal data to create systems that are more robust or have higher accuracy when performing certain tasks. However, information fusion algorithms face several obstacles. Questions like when to fuse information or even how to deal with noisy, missing or unaligned data might arise. Typically, fusion techniques are classified regarding the moment when that fusion occurs. Early fusion, that fuses raw data before feeding it to the neural network, middle fusion, where fusion occurs, possibly more than once, at different depths of the network, and late fusion, that combines the output of multiple networks of different modalities. No matter the moment where data is fused, the goal is to be able to employ data-driven methods to a task that used to require a tremendous amount

of work and thorough sensor calibrations when relying on classic fusion implementations. DL techniques make, therefore, an interesting solution to the sensor fusion problem, relying on data to build knowledge, instead of relying on handcrafted algorithms.

1.3 Objectives

Relying on the tremendous advances that Artificial Intelligence (AI) has seen over the years, the scope of this work focuses on the development of an robust odometry system for maritime environments, using DL techniques. The main goal is to employ data-driven methods to fuse information from different data sources in a late fusion fashion to improve the localisation task of ASVs in harbor and port environments, taking advantage of the knowledge and geometric approaches, and circumventing the need of very large training datasets. The work focuses on situations where GPS/INS systems lack consistency, which can happen, for instance, when an ASV approaches a structure to perform Operation and Maintenance (O&M) missions. Therefore, the developed work can be decomposed on several different objectives, namely:

- The evaluation of well-established odometry estimation algorithms under maritime conditions, and the discussion of the strengths and vulnerabilities of each of one of the algorithms under assessment;
- The development of a robust odometry estimation algorithm, based on DL techniques, that combines multi-modal information from different sensors, namely, visual cameras, LiDAR, and GNSS/INS systems, taking into account the results of the assessment referred to in the previous point;
- The proposal and deployment of an autonomous navigation algorithm, based on a unified and flexible multi-layer software architecture, used for data collection in an autonomous fashion;
- The collection of a multi-modal dataset in a real environment, using ASVs. The dataset has to provide a truthful localisation estimation, making use of high-accuracy sensors, used for ground-truth purposes;
- The evaluation of the proposed algorithm's results on the dataset collected in a real-world environment. This means the algorithm is to be tested with signals that are prone to failures, simulating non-ideal and real-life scenarios.

1.4 Scope of the Work

The work carried out throughout this dissertation was done under the H2020 Project ATLANTIS [3], a project for the development and testing of robotic technologies for inspection and maintenance

[3] https://www.atlantis-h2020.eu/

of offshore wind farms. This work also contributed for the article "ATLANTIS Coastal Testbed: A near-real playground for the testing and validation of robotics for O&M" [13], published as part of the OCEANS Limerick 2023 conference proceedings.

1.5 Document Structure

The present document is divided into five main chapters. Chapter 1 introduces the scope of this dissertation, the relevance of VO methods, and the importance of fusing sensor data in the context of autonomous maritime navigation.

In chapter 2, an overview of the present state of the art of VO implementations and data fusion techniques is presented, for both classical and Deep Learning approaches.

Chapter 3 covers the work carried throughout the course of this dissertation, ranging from the collection of an odometry estimation-focused dataset and the preparations that lead to it, to the benchmark, in maritime scenarios, of odometry estimation systems that rely on different modalities, and also presenting an autonomous navigation algorithm proposed for autonomous data collection tasks. Finally, a self-attention-based fusion system is proposed. In chapter 4, the results of that multi-modal system are evaluated and discussed.

Chapter 2

Related Work

This chapter presents the state of the art of odometry estimation algorithms, covering the most recent advances and major works the the research field of localisation estimation systems. With a particular emphasis on visual-based methods (section 2.2), it covers classic or geometric approaches and more recent data-driven algorithms. The scarce application of VO in maritime environments is also presented, as well as the most conventional GPS/INS localisation systems usually adopted by ASVs. Then, existing multi-modal odometry algorithms are presented (section 2.3). Finally, the content covered in this chapter undergoes a critical review in section 2.4, concentrating on the weaknesses of the existing pose estimation systems, highlighting the obstacles that the fusion model presented in section 3.4 must circumvent.

2.1 Background

Odometry is a dead-reckoning technique to estimate the change of position and orientation over time. By norm, the pose of an agent is incrementally calculated with respect to its initial position. The concatenation of motion estimates between sensor measurements is what allows for a relative pose calculation. The use of wheel encoders is one of the simplest approaches to the odometry problem, but a large number of distinct sensors might be adopted. To name a few, LiDAR, Radar and visual camera-based odometry systems have been explored for decades. Egomotion is, by definition, the 3D motion of a system relative to its environment. The process of estimating the egomotion of an agent using single or multiple cameras as input is called VO, whose formulation does not differ much from classical odometry estimation methods. The idea is to iteratively calculate the displacement of a sensor between different measurements (fig. 2.1). In the case of a VO system, that displacement is calculated based on the characteristics of consecutive image frames. Therefore, the sequential reconstruction of the path followed by an agent is made by concatenating the displacements over time. Those displacements can, sometimes, be wrongly estimated, leading to error accumulation, i.e. drifts. An optimisation step can be performed to refine the pose calculation based on a fixed number of previous image frames. In spite of being an area that has been

extensively studied over the years, VO and V-SLAM methods are still not widely used in ASVs as their performance is largely affected by marine environment constraints.

To recover from drifts and error accumulation, typical in VO, an external data source can be used. GNSS makes an interesting solution to that problem, as these sensors estimate absolute position, unlike dead-reckoning methods. The fusion of absolute and relative pose estimators might increase the robustness of a localisation system. Relative pose estimators could benefit from high-rate execution cycles, making up for the general GNSS' low sample rate or even GNSS signal outages.

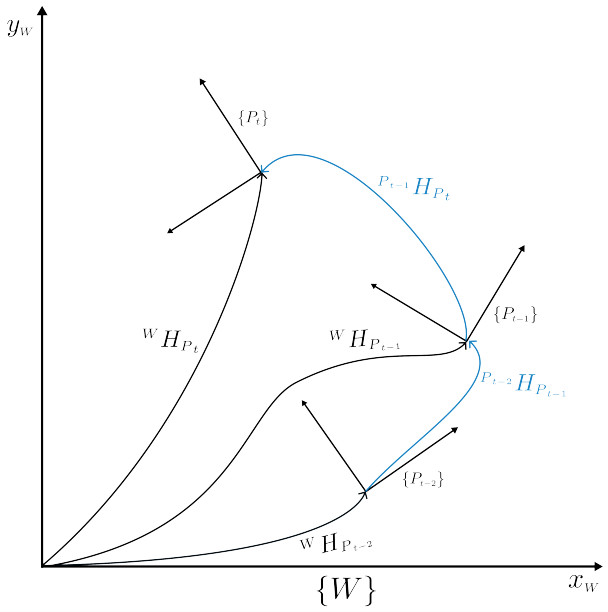


Figure 2.1: In this figure, three two-dimensional poses (P_{t-2}, P_{t-1} and P_t) are represented on a global frame (W). These poses can be expressed in relation to the global frame, as global poses (represented by black arrows), or relative to each other, as relative poses (represented by blue arrows). Dead-reckoning methods, as it is the case of most of VO methods, work by the principle of iteratively estimating relative poses between two measurements. An example of a localisation estimation method that calculates position relative to a global frame (absolute poses) is a GNSS/INS system, that references position and orientation relative to the planet earth.

Visual SLAM (V-SLAM) implementations go even further on the error accumulation issue by simultaneously creating a map of the environment. When an area that had already been mapped is revisited, a loop closure can be detected, reducing error accumulation. Despite making the localisation task more accurate, V-SLAM methods are usually computationally more expensive than VO systems [14].

2.2 Visual Odometry

The complexity of VO and V-SLAM systems has been growing over the years. Leveraged by the development of computer vision, combined with the increasing performance of computational hardware, these systems have been presenting solid results in some environments and conditions. There are, however, several different possible approaches when dealing with vision-based localisation systems, whether data is acquired by a monocular or stereo camera, or data processing is made via traditional computer vision methods or via AI methods (e.g. DL approaches).

2.2.1 Geometric Approaches

Geometric approaches estimate the egomotion of a camera by applying handcrafted algorithms to a sequence of images and can be divided into feature-based methods, appearance-based methods or hybrid methods (fig. 2.3).

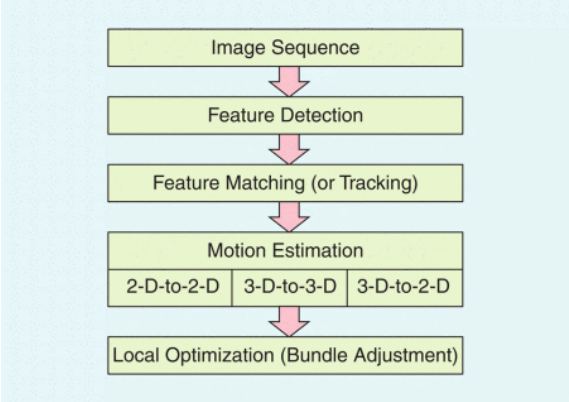
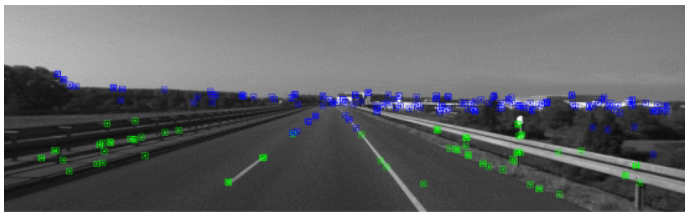


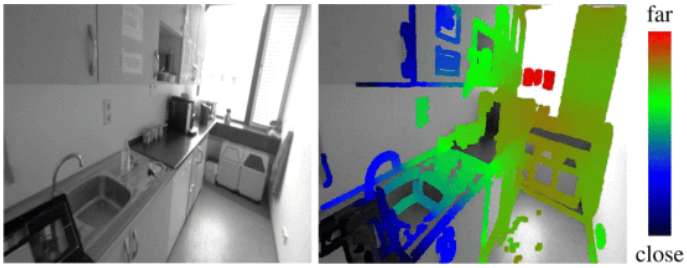
Figure 2.2: A typical Feature-based Visual Odometry Pipeline [14]

To better understand why the resort to VO is still not commonplace in maritime environments, a succinct description of a visual odometry system's pipeline is presented (fig. 2.2). Put simply, an image is captured and analysed. That analysis, which can be preceded by some image pre-processing, consists of identifying distinct features. Knowing the camera's intrinsic parameters (focal length, optical center, and skew coefficient) allows points of those characteristics to be mapped into a three-dimensional space. The same search for features is done as a new image is captured. Comparing those with the ones extracted from the preceding image makes it possible to perceive the change of position in the two-dimensional image plane and thus in a three-dimensional space. Interactively calculating that motion is what makes it possible to sequentially reconstruct the path followed by an agent.

Features, however, are sometimes difficult to extract and match (or track). On top of that, they must also be fixed points in the 3-D space, since a moving agent is able to perceive its motion when it has some fixed references to rely on - just like the human perception of motion through visual input. Hence, visual odometry systems should, ideally, only rely on fixed features, since the ones that are not fixed lead to motion estimation errors. Some VO implementations break the



(a) ORB-SLAM [15] features, extracted and tracked using a feature-based approach to the VO problem. Green features represent points that are close enough to the visual cameras, while blue features are too far to be considered by the algotrithm.



(b) Semi-Dense Visual Odometry for a Monocular Camera [16]. Instead of extracting distinct visual features that can be tracked over time, this appearance-based approach minimises the photometric error over image frames, in regions that have a non-negligible image gradient

Figure 2.3: Example of a feature-based (top) and an appearance-based (bottom) method.

pipeline described above by, for instance, using the intensity information of all pixels of a sequence of image frames and minimising photometric errors between image frames. More recently, the scientific community started turning its attention to Deep Learning techniques to regress visual camera poses.

Feature-Based methods

Feature-Based methods rely on distinct characteristics of image frames, which might consist of edges (boundaries between image regions with distinct properties), corners (interception of edges), blobs (image areas with pixels that share similar properties), etc., and are commonly referred to as features in the computer vision field. These features are tracked or matched between image frames and the difference in position of those features is what makes it possible to calculate the camera motion in the 3D space. Features have properties that make them appealing to the design of geometric VO models. They are local and, for that reason, robust to occlusions, can be efficiently extracted and usually appear in huge numbers in a single image. On top of that, images of the same scene tend to have repeatable features, even when subject to different viewing conditions. A plethora of feature extraction methods are currently available. Given the characteristics of feature-based VO methods, the most common types of feature extractors for motion estimation rely on corner or blob detectors. After the extraction process, a feature correspondence has to be

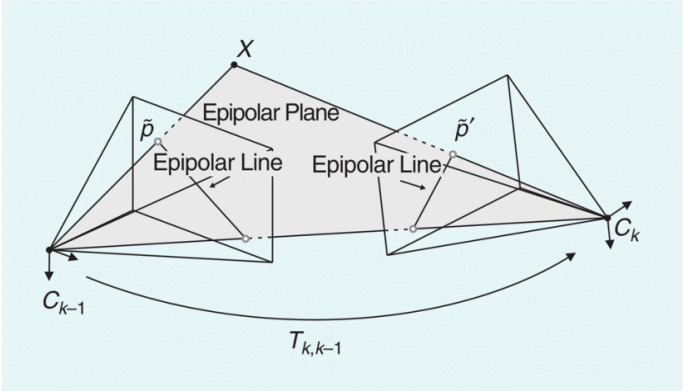


Figure 2.4: Epipolar geometry (image from [14]). Point X in a 3D world frame is projected onto two consecutive image frames as $\tilde{p}$ and $\tilde{p}'$. Notice that dragging X along the line segment that connects X to the origin of C_{k-1} does not change the coordinates of $\tilde{p}$ on the image frame, but drags $\tilde{p}'$ along a line on the image frame from time instant k. That line is the epipolar line. This geometry principle is what makes it possible to extract the essential matrix and, therefore, the translation and rotation between two image frames.

performed to calculate motion. Correspondence can be calculated directly from the image features in the 2D image planes (2D to 2D), from a 3D structure (3D to 3D) or from a 3D structure to an image plane (3D to 2D). The 2D to 2D approach estimates motion directly from the change in position from correspondent features in two consecutive frames in time. The essential matrix, encapsulating a rotation and a translation between two camera frames is estimated using epipolar geometry (fig. 2.4). The 3D to 3D methods project features from the image planes into a 3D space, usually thanks to stereo camera pairs, estimating motion and its scale based on the triangulation of features. Lastly, 3D to 2D correspondence methods have been proved to be able to produce better results when compared to the previous implementations [14]. 3D feature points are reprojected onto the following image's 2D plane, reducing the reprojection error of features.

Mono-vo [17], a monocular solution for the VO problem, uses the FAST [18] corner detector to extract features, and a Kanade-Lucas-Tomasi [19] feature tracker to find feature correspondences. The scale ambiguity problem, common to monocular approaches, is solved using an external data source. In order to solve the scale ambiguity problem, TGVO [20] uses the output of a calibrated stereo camera to estimate motion. Images are divided into patches, or buckets, and the number of features in each bucket is reduced to decrease complexity and increase the performance of the algorithm. RANSAC [21] is then applied to reject feature outliers extracted from dynamic objects. Lastly, a Kalman Filter is used to estimate the state of the dynamic system. Regarding V-SLAM implementations, ORB-SLAM [15, 22, 23] uses the ORB feature detector [24], an open-source feature extractor and descriptor. ORB-SLAM allows for both monocular, stereo and RGB-D camera systems and performs a real-time mapping of the environment, while also estimating the odometry of an agent. A loop-closure detection is used, relying on a bags of words place recognition module, improving the localisation estimation when a local is revisited.

Appearance-Based methods

Unlike feature-based methods, appearance-based implementations use pixel-intensity information from the entire image. Not only are they computationally more expensive when compared to feature-based methods, but they also tend to lack some robustness, due to their susceptibility to brightness variations. Nevertheless, some researchers have explored the potential of such methods, like it is the case of DSO [25].

DSO uses images from a monocular camera and applies a direct method to sample points, not only with distinct characteristics, like features, but also with points that have weak intensity variations when compared to their neighbors. This way, the method relies on more information than feature-based methods, but ignores some points to decrease computation time without loss of performance. It is heavily dependent on calibrations, namely a geometric and a photometric calibration, and the use of rolling shutter cameras is not recommended. A stereo implementation of the method has also been proposed by Wang et al. [26].

LSD-SLAM [27] is an example of a V-SLAM monocular appearance-based implementation. Like DSO, LSD-SLAM tries to minimise the photometric error between image frames and a given key frame. This approach differs from geometric implementations, where the goal is to minimise the geometric projection error of features.

2.2.2 Deep Learning Approaches

The first DL architectures to be developed for pose estimation purposes learned via supervised learning methods. PoseNet [28] was the first implementation of a Convolutional Neural Network (CNN) that could regress the pose of a camera from monocular visual information. The neural network uses pre-trained weights from a CNN developed for classification purposes and, aided by transfer learning, is able to make predictions even in harsh conditions, such as poor light and dynamic objects, and with unknown camera intrinsic parameters. The work was later refined by using a loss function based on the geometric projection error from the predicted pose [29]. PoseNet (fig. 2.5) does not consist, however, of an odometry estimation implementation. For that matter, the pose is regressed from a single image frame and not from a sequence of images. In the work proposed by Clark et al. [30], the pose is estimated from a stream of temporally aligned images, improving the results attained by PoseNet, while the core of the CNN remains the same. That is, in fact, the path followed by most VO systems developed with DL techniques. Making use of temporal dynamics, such implementations rely on Recurrent-CNN (RCNN) models to retain information in between image frames. DeepVO [31], for instance, estimates the camera motion between two consecutive raw monocular images, consisting of an end-to-end VO system. The network receives two stacked images as an input, extracts features via a CNN, inspired by the work by Dosovitskiy et al. [32] and their proposed network for optical flow estimation, and feeds them into two stacked Long Short-Term Memory (LSTM) layers, that output the final pose. The network presented results compared with monocular feature-based implementations, but produced

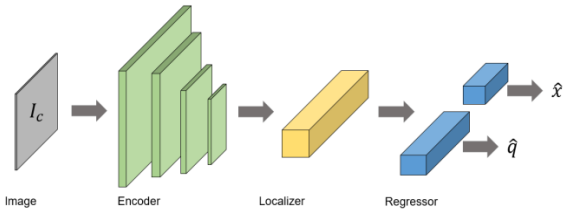


Figure 2.5: PoseNet [28] architecture. Image courtesy of [37]. The encoder extracts features from images, using pre-learned convolution operations, and poses are regressed in subsequent layers. In PoseNet, orientation is represented in the form of unit quaternions.

high translational errors at high speeds. Highway scenarios also represented difficult environments for the networks, due to the lack of features and the large number of moving objects. Wang et al. [33] proposed a probabilistic VO system. In addition to the estimation of the pose, the model also regresses the uncertainty that the said estimation might carry. The authors pointed out that those uncertainties could be used to produce sensor fusion-based odometry systems, such as the fusion of a VO system with GPS measurements (see section 2.3.2). The work by Valada et al. [34] focused on reducing the drift that usually comes from the error accumulation of odometry systems. Two separate sub-networks regress both egomotion and a global pose, sharing weights so that the global pose estimation task can be improved via deep auxiliary learning. The inputs of the network consist of two consecutive monocular images, and the final goal of the architecture is the absolute location estimation. In other words, the odometry network aids the global localisation network to by sharing information. Using a similar line of thinking, but shifting the target to the odometry task, Lin et al. [35] achieved promising results using two separated RCNNs. The outputs of the VO and the global pose network are then fused via fully-connected fusion layers, improving the odometry estimation task by a great margin when compared to DeepVO [31] on the KITTI benchmark [36], a well-established dataset in the computer vision and autonomous driving community.

Even though supervised DL approaches to the VO problem have proved to be able to achieve good results, it is usually necessary to label huge amounts of data for training purposes, which is a weary task. Thus, unsupervised learning comes into play, aiming to release people from such work. Ummenhofer et al. [38] made way to such learning problems, with an architecture that could learn depth and motion from two images without any supervision signals, being the structure from motion the main goal of their work. The major leap forward in unsupervised methods was made with SfMLearner [39]. The work consists of two distinct sub-networks: one calculates a depth image from a single monocular image and the other, aided by the depth image, learns to estimate motion from a stream of five images. The goal is to minimise the photometric error of a projection from a target image frame to another nearby image frame. Additionally, an explainability mask is estimated, so that dynamic objects and occlusions can be ignored by the photometric loss during training. GeoNet [40] also used several distinct sub-networks, similar to SfMLearner [39]. However, in GeoNet, occlusions and dynamic objects are treated differently. Depth and motion

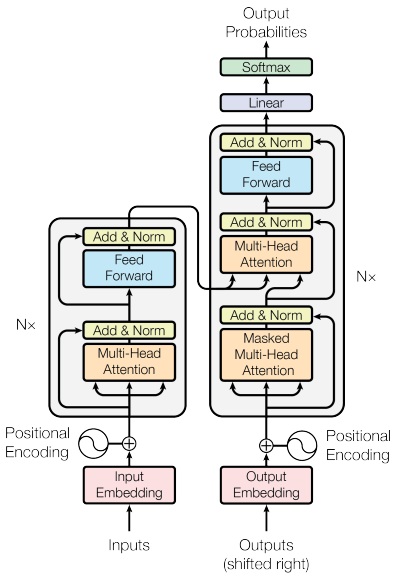


Figure 2.6: Transformer Network Architecture [43]. Originally developed for Natural Language Processing (NLP) tasks. The encoder (left block) extracts features from an input sequence and feeds them into the decoder (right block), where self-attention operations merge information with previous outputs of the network.

information is estimated by two sub-networks and then fused, producing the rigid flow, which is then cascaded by a network that estimates object movement relative to the world plane. In UnDeepVO [41], the absolute scale recovery is done using a depth map, which is trained by a set of stereo images. During test time, however, only a monocular image is required for the estimation of both depth and pose information. It is worth noting that all of the aforementioned unsupervised methods depend on the information given by the camera's intrinsic parameters. Those compose the intrinsic matrix, which usually has to be computed via a calibration process and defines the projection of 3D points onto a 2D image plane. To circumvent that calibration process, a few authors have explored the ability of data-driven models to predict the intrinsic matrix, like is the case of the work by Chen et al. [42]. The training phase relies on loss functions that capture geometric constraints and photometric consistency, while using solely monocular images. The intrinsics are optimised by a multi-view 3D structure consistency loss, by reprojecting a pixel from a source image frame to a target frame.

In 2017, the introduction of Transformers networks [43] (fig. 2.6) revolutionised DL algorithms, with focus on textual structured data. Furthermore, these models have been showing their ability to perform well in tasks that are usually carried by CNNs, such as image recognition tasks. VO algorithms have also been implemented, as in the case of the work by Li et al. [44]. The Transformer architecture is able to model geometry and temporal information over a short period of time and to regress a 6-DoF pose in an unsupervised fashion. The transformer model's results are, however, not as good as the results obtained by the other architecture proposed in the same publication, which uses optical-flow predictions to estimate pose variations. In fact, the architecture is

mainly used in collaborative training of a more complex model. The overall architecture outperformed other unsupervised implementations, but performed poorly in situations where a sequence of image frames represent a static camera, with no variation in pose.

2.2.3 Visual Odometry in ASVs

Following the path of autonomous driving, where visual information is often used for situational awareness and odometry estimation [45], some investigation has been done, studying the potential application of visual cameras in autonomous or unmanned boats and ships. Volden et al. [46] used fiducial markers as guidance to the autonomous docking maneuver. The detection of such markers is done using a previously trained CNN. Both a stereo and a monocular implementation were tested, with the latter achieving the best results. The use of markers is not ideal as it requires modifications to the environment where the autonomous agent might operate, making it dependent on external factors. However, the examples in the literature of marker-free visually assisted navigation techniques is still very scarce. Regarding visual odometry or visual SLAM techniques, Kriechbaumer et al. [47] performed an evaluation of two stereo geometric VO implementations in inland waters, using feature and appearance-based methods, with the best results produced by the feature-based VO implementation. Still, the results were not good enough, as the error accumulation in long trajectories assumed very large values. Wang et al. [48] compared the behavior of ORB-SLAM [22] against DSO [25] in harbor conditions. They concluded that ORB-SLAM performed poorly under a large-scale harbor scene, where a big portion of the image is occupied by the sea surface, thus not being able to extract robust features. On the other hand, in spite of having produced better results for the odometry task, DSO is not able to perform loop-closing detection, unlike the feature-based method, taking its toll on position estimation results over time.

2.3 Multi-modal Odometry

Fusing classical VO systems with other sensors' measurements is an area of investigation that has been intensively studied over the past years. From classical approaches to data-driven methods, a plethora of algorithms is available as of today. Classical methods are usually divided into loosely and tightly-coupled approaches, whereas DL methods are normally divided into early, middle or late fusion techniques, as described below.

2.3.1 Classic Fusion Methods

Sensor fusion techniques have been employed for decades, like it is the case of the use of the Extended Kalman Filter (EKF) to perform position and orientation estimation of non-linear systems, coping with noisy measurements and estimation errors. A common practice of such EKF applications is of the fusion of GPS and inertial measurements for outdoor and maritime applications [49]. The filter fuses position and orientation information from a GNSS and an IMU, respectively, relying on uncertainty estimates, expressed by covariance matrices. Kalman Filters and its variations

rely, however, on Gaussian sensor models, which might not always be a truthful representation of a sensor. In order to deal with non-Gaussian models, Monte-Carlo methods, such as Particle Filters, come up as an alternative [50], despite being computationally more expensive.

In maritime navigation, the fusion of GNSS and inertial measurements still makes up for the lack of better (and cheaper) systems. The use of the well-known EKF is the most common approach [5, 51]. In fact, even different approaches usually rely on variations of the Kalman Filter framework as the backbone of the GPS/INS-based localisation method, like it is the case of the work by Naeem et al. [52], who proposed a Fuzzy logic adaptive Federated Kalman Filter. The goal was to detect and isolate faults from the individual sensors, adding robustness to the navigation task. Romanovas et al. [53], further exploited Kalman Filter's capabilities, by adding a third sensor, a Doppler Velocity Log, to a GPS and IMU system, still taking advantage of a non-linear fusion filter when the vessel is subject to GPS outages, even when those failures last for a long period of time.

With highlight on visual-inertial odometry systems, driven by the potential applications in UAVs, different types of fusion methods have been developed. The goal is to fuse the information of a proprioceptive sensor - an IMU - and an exteroceptive one - a camera. Visual-inertial odometry systems can be divided in loosely and tightly-coupled approaches. The former use IMU measurements to aid a main odometry estimation modality. The latter fuse information from the sensors, frequently resorting to factor-graph approaches. Most loosely-coupled methods are filtering-based fusion techniques. Filtering-based approaches commonly resort to EKFs, using IMU measurements for the filter's prediction phase and visual features to compute the update steps [54]. Although filter-based approaches tend to use less computational resources when compared to optimisation-based (or tightly-coupled) approaches, they are usually outperformed by the latter. That happens because optimisation methods estimate pose by optimising key information from images and inertial measurements. VINS-Mono [55] is a well-established implementation of such approaches. An initialisation process loosely fuses IMU and camera measurements in order to recover the absolute scale of the movement. Then, a tightly-coupled non-linear optimisation problem is solved over a bounded sliding window containing measurements from both sensors. Additionally, a loop closure detection module is proposed, which increases the complexity of the localisation process, but also increases the overall accuracy of the system. Building upon previous implementations, ORB-SLAM3 [23] is able to perform fusion between visual and inertial measurements. The implementation works on monocular, stereo and RGB-D camera systems and outperforms most classical visual-inertial odometry systems in different scenarios.

Other types of sensors have also been in use over the past years. LiDAR sensors, for instance, have been gaining popularity thanks to the high precision ranging measurements, based on the time-of-flight principle of light beams. The price of such sensors can still have a deterrent effect on many applications, but it has not diminished the interest of the scientific community. V-LOAM [56] modifies previous works from the same authors in order to fuse a visual odometry system

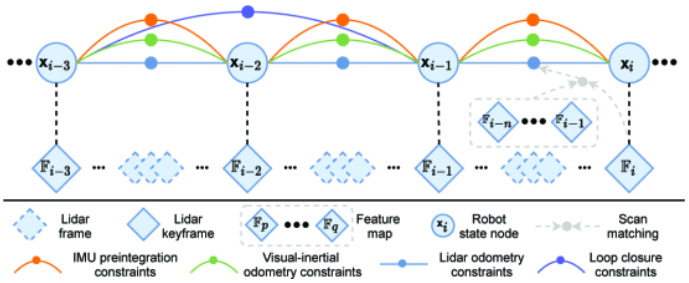


Figure 2.7: Factor-graph from LVI-SAM [57]. The approach is formulated as a maximum a posteriori (MAP) problem, solved by jointly optimising constraints from IMU preintegration, visual and lidar odometry, and loop closure in a factor graph.

with a LiDAR odometry implementation. Visual odometry estimates change in motion at high frequencies (60Hz), aided by depth information acquired by a LiDAR. Those LiDAR measurements are also used to refine the visual odometry estimates at a much lower rate (1Hz). This way, robustness is added to the visual odometry module, which can lack some consistency in low features environments or low-light situations. Going even further, Shan et al. [57] fused a tightly-coupled visual-inertial odometry system with a tightly-coupled LiDAR inertial system (fig. 2.7). The result is a tightly-coupled Lidar-Visual-Inertial Odometry method that fuses data from three different sensors. To increase the robustness of the whole system, the method is able to function even when one of the sub-systems fails.

Concerning sensor fusion techniques for autonomous crafts with odometry purposes, not many publications are available, as GPS/INS odometry is still the standard for maritime navigation. An exception to that was the paper by Leedekerken et al. [58], where a SLAM implementation is proposed for navigation in conditions where GPS signal might lack some consistency. The mapping of the environment is made both above and below the water surface, using measurements from a LiDAR and a Sonar sensor. Both modalities are projected into a voxel tree and, then, submaps acquired during short periods of time are matched against a local map via Iterative Closest Point (ICP).

2.3.2 Deep Learning Methods

Combining sensor fusion and DL has led to the development of robust learning-based odometry systems. Similarly to classical approaches, visual and inertial information make an interesting combination for such systems. VINet [59] was the first DL implementation of a visual-inertial odometry system, fusing the output of a previously trained CNN and the measurements of an IMU. As visual and inertial sensors usually have different operational frequencies, the authors used an LSTM unit to store information from a sequence of IMU measurements. The information is later fused using a Core-LSTM, trained in a supervised fashion. Similar implementations have been tested in environments that represent harsh operational conditions, like it is the case of underwater navigation [60]. Almalioglu et al. [61] enhanced the visual-inertial system with an unsupervised implementation of the odometry estimation. On top of that, a set of measurements

from the IMU is organised in a 2D matrix and then processed via a CNN, extracting features from those measurements. The features are then fused with the ones extracted from images and the agent's pose is regressed. Using a very different architecture, Wan et al. [62] used learning-based sub-systems to regress the pose from each individual sensor and then fused the information with a classical method, using an EKF.

VLONet [63] fuses LiDAR point clouds with visual data to estimate motion in an unsupervised way. The 3D points are projected onto a 2D frame, having the camera frame as a reference, providing depth information to the system. The network is based on an encoder-decoder structure and the training phase relies on a siamese architecture that calculates losses as a function of the flip consistency of both depth and pose predictions. Also using a LiDAR sensor, but fusing it with inertial measurements instead, Javanmard-Gh et al. proposed DeepLIO [64]. To structure the point cloud data, points pass through a spherical projection. Two sub-networks independently extract features from both modalities, which are then fused using an attention-based soft-fusion method.

Some probabilistic models have also been explored. Apart from pose, uncertainty might also be estimated, making it possible to predict motion with more confidence from the information extracted from each modality. Kaygusuz et al. [65] fused pose predictions from the data of multiple cameras. Image features are extracted and then fed into an MDN [66], which regresses pose transformations and uncertainty in a supervised and unsupervised way, respectively. A fusion module consisting of Multilayer Perceptrons and an RNN regresses the final 6-DoF motion estimation. Also using density estimates, Pillai and Leonard [67] experimented a learning-based VO system fused with intermittent GPS updates, in order to recover from drift errors that come with dead-reckoning systems like it is the case of odometry estimation.

These DL approaches are, however, still highly dependent on very large datasets, given the high capacity of CNN-based architectures. On top of that, most of these works did not result in open-source repositories, making it very hard to assess the performance of such implementations in custom datasets.

2.4 Critical Review and Challenges

In the previous sections, some of the most relevant works in the VO research field were presented. An overview of methods that fuse visual information with other modalities for localisation purposes was also done, showing there are many different approaches when dealing with images for leveraging the performance of pose estimation systems. This section casts a critical look at those different approaches, always taking into account the near-the-shore environment where the work of this dissertation has been conducted.

If it is sure, as it has been demonstrated previously in this chapter, that VO is yet to be considered a viable solution for USVs' usage, there is also a gap to be filled regarding research in this area.

The lack of publicly available datasets with a focus on visual (and LiDAR) data makes it very hard to deploy odometry estimation systems on bodies of water. Still, the results achieved by different techniques on publicly available datasets, like the KITTI [36] dataset, offer a glimpse of what is to be expected from visual and fusion odometry systems in certain conditions.

A first glance through the top spots of the KITTI dataset evaluation page [1] expose the lack of end-to-end data-driven odometry methods. Indeed, visual DL methods that have been able to achieve reasonable performance are those that fuse geometric knowledge with learnable parameters (e.g. DVSO [68], where deep depth predictions are incorporated into an appearance-based VO method [69]). As to the reason why DL approaches still lack performance when compared to classical ones, several factors might come into play. First, the lack of availability of the so much needed training data, where a lot of different scenarios are covered, is certainly the major drawback. Second, classical approaches have been growing in robustness and consistency in specific environments and conditions, making these methods very challenging to surpass performance-wise.

Regarding VO classical approaches, a huge problem arises when dealing with images captured above the water: most of the area of the image represents the water itself, or even the sky. Unlike driving environments, where features extracted from the ground plane are reliable, those extracted from maritime scenarios are prone to reflections and distortions of the water. Weather conditions also have the potential to further unveil the weakness of VO systems. Undulation, for instance, imposes fast motion and perspective changes, which can severely distort images, affecting feature extraction and/or tracking. Appearance-based approaches are also prone to erroneous photometric error estimation between images, as a significant portion of those images is not static. Adding to that, most of the time, navigation is made in textureless places (e.g. offshore), making it impossible for VO systems to operate.

Finally, fusion algorithms have been demonstrating their ability to achieve superior performances, extracting from different sensors' signals their strongest qualities, but they are still far from being perfect. Classical techniques are usually dependent on the finest calibration procedures, which take time and resources. On top of that, disturbances in the environment and, therefore, in sensors, might severely degrade the quality of those calibrations, and the sea is known for being very harsh on materials. Besides that, when operating near structures, GNSS/INS measurements tend to degrade. GPS signals can suffer from reflections, and magnetic fields can be distorted by ferromagnetic structures, usually present near harbors. These effects are not captured by sensor calibrations. DL stands as a solution to those problems. It totally or partially circumvents the necessity to perform thorough extrinsic calibrations while still getting the best of each modality to be fused. Not only can DL methods fuse different sources of information, but they also can leverage handcrafted and robust classical algorithms with knowledge extracted from data. That is the direction this work is heading to over the following chapters, with a focus on the performance

[1] https://www.cvlibs.net/datasets/kitti/eval_odometry.php

of such odometry estimation algorithms in maritime environments. On top of that, a late fusion approach is used, reducing the capacity of the proposed model, meaning smaller training datasets can be used to achieve good performances.

Chapter 3

Self-Attention Multi-Modal Odometry

The present chapter describes the work on the gaps mentioned in chapter 2. For that, a dataset had to be collected under real-life conditions. The problem to be addressed is formulated in section 3.1. In section 3.2, the scenario where data was acquired is described, as well as the used vehicles and their preparation. Then, in section 3.2.1, a modular software architecture is proposed, which was later used for autonomous data collection. Section 3.3 is a benchmark of existing and well-established odometry systems, some of which are later selected as the input of a self-attention-based fusion network (section 3.4).

3.1 Problem formulation

A rigid-body transformation, ${}^{A}H_{B} \in SE(3)$, relates the position, orientation or pose of an object from the perspective of frame A to that of frame B. ${}^{A}H_{B}$ comprises both a rotation, ${}^{A}R_{B} \in SO(3)$, and a translation, ${}^{A}t_{B}$. Notation usually makes use of the form of homogeneous matrices and vectors such that:

$$\underset{(4\times4)}{{}^{A}H_{B}} = \left[\begin{array}{c:c} \underset{(3\times3)}{{}^{A}R_{B}} & \underset{(3\times1)}{{}^{A}t_{B}} \\ \hdashline 0 \;\; 0 \;\; 0 & 1 \end{array}\right], \tag{3.1}$$

Meaning a point p, from the perspective of frame A, ${}^{A}p$, can be obtained, as:

$$\underset{(4\times1)}{{}^{A}\tilde{p}} = \begin{bmatrix} \overset{A}{p} \\ \underset{(3\times1)}{} \\ 1 \end{bmatrix} = \underset{(4\times4)}{{}^{A}H_{B}} \underset{(4\times1)}{{}^{B}\tilde{p}}\ , \tag{3.2}$$

, where $\tilde{p}$ stands for the homogeneous representation of p Additionally, ${}^{B}p$ can be obtained from ${}^{A}p$, given the transformation from A to B as:

$${}^{B}\tilde{p} = {}^{A}H_{B}{}^{-1} \cdot {}^{A}\tilde{p} = {}^{B}H_{A} \cdot {}^{A}\tilde{p}, \tag{3.3}$$

These are the foundations of the odometry estimation problem that is covered in section 3.4. Other representations of rigid-body transformations are valid. In fact, computations with rotations are normally made making use of unit quaternions. A unit quaternion, q, is a four-dimensional data structure, with one real part and three imaginary components, $q = \mathbf{w} + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$. The imaginary components define an axis of rotation, while w gives the magnitude of the rotation around that axis. A simpler rotation representation resorts to *Euler angles*. *Euler angles* may be extrinsic (rotations about an original and fixed frame), or intrinsic (rotations about the axes of the rotating coordinate system). In mobile robotics, the former is of standard usage, usually referred to as $roll(\phi)$ (rotation about the x axis), $pitch(\theta)$ (rotation about the y axis) and $yaw(\psi)$ (rotation about the z axis). There is, however, a major drawback concerning this last representation of orientation, as singularities might occur. For that reason, the usage of *Euler angles* is not encouraged. Either way, there is a direct relation between any of the conventions described above [70], and in this document, there are references to all three of those conventions.

Let ${}_{i}H$ be a set of the last n pose transformations of a set of odometry estimation systems, $i \in S = \{1, 2, ..., s\}$, so that ${}_{i}H = \{{}^{0}_{i}H_{1}, {}^{1}_{i}H_{2}, ..., {}^{n-2}_{i}H_{n-1}, {}^{n-1}_{i}H_{n}\}$. The fusion system can be expressed as:

$${}_{f}H = f({}_{1}H, {}_{2}H, ..., {}_{s}H), \tag{3.4}$$

where ${}_{f}H$ is the set of the last n pose transformations, as estimated by the fusion system. This means that the k'th pose of an agent can be computed, through composition, and relative to the first estimation, as:

$${}^{0}H_{k} = \prod_{i=1}^{k} {}^{i-1}H_{i}, \tag{3.5}$$

3.2 Data collection: building a real-world maritime dataset

Chapter 2 laid bare the biggest barrier to the development of a data-driven fusion algorithm: the lack of available data collected by vessels. The course of action of the present work started, therefore, with real-world data collection. In this regard, data was acquired in the ATLANTIS Coastal Testbed, in Viana do Castelo (figure 3.1), where the available infrastructure creates the conditions for testing and simulating O&M activities in offshore wind farms. The testbed provides a wide and dynamic scenario, subject to the navigation of a variety of watercrafts, and is equipped with a floating structure that resembles that of an offshore wind turbine. Hence, this work was performed, in its entirety, under realistic maritime conditions, with real ASVs, described ahead.

Figure 3.1: The ATLANTIS Coastal Testbed, in Viana do Castelo, and the floating structure, DURIUS.

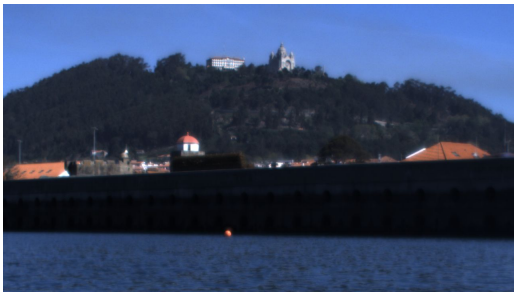
Given the ample scene provided by the ATLANTIS testbed and the exposure to harsh maritime conditions, the predicted barriers to the acquisition of visual data are confirmed. Figure 3.2 depicts some of the challenges to be overcome. While navigating on the testbed, the ASV has to deal with images of non-static objects, such as floating platforms and watercrafts, which trigger a misleading motion perception. In some situations, the only static and reliable objects are very far from the place where the ASV might be navigating (distances superior to 200 meters). In others, luminance conditions, natural of an outdoor environment, can take its toll in the performance of VO algorithms.

For data acquisition, two ASVs were used. The first, the SENSE ASV (fig. 1.1), was used for initial tests and data collection. SENSE is a catamaran-based craft, suitable for operations in nearly flat water conditions. Then, a more robust craft, Nautilus (fig. 3.3), served the purpose of the majority of the work. Nautilus has a monohull configuration that allows it to perform under moderate sea state conditions. Both vehicles carry a variety of sensors, suitable for O&M activities.

Focusing on the array of sensors available on Nautilus 3.4, there are available:

(a) Features of non-static objects.

(b) Features too far from the ASV.



(c) Harsh luminance conditions.

(d) Textureless image.

Figure 3.2: Testbed harsh conditions to VO.

- Two visual cameras: Bandwidth limitations imposed that the image resolution was set to 1280x720 pixels, and the frame rate to 10 frames per second. The image format was also changed due to the available bandwidth. An 8-bit Bayer filter reduces the dimensions of the data packets during the transmission of information, hence reducing the load on the networking hardware. An external triggering software was developed, allowing the cameras to acquire synchronized images. This means the camera assembly can work as a stereo perception module, with a baseline of 50 centimeters;

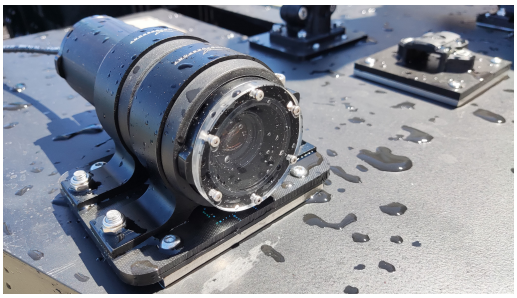
- A 128-channel LiDAR: Extremely precise, a sensor that makes use of the time-of-flight of light beams principle, and allows for accurate ranging measurements over a 360º field of view (FoV). Like the cameras, LiDAR measurements are made at a frequency of 10Hz;

- An autonomous navigation module: Apart from the control module available, which is not used for the purpose of this dissertation, this autonomous navigation module is equipped with an IMU and a GPS, that output position and attitude measurements relative to an on-earth fixed referential;

- A Real Time Kinematic (RTK) GPS: Allows for centimetric accuracy position measurements, relative to a fixed base station. Used for the generation of the ground-truth signal;

- A 9-DoF IMU: Outputs orientation, angular velocity, and linear acceleration information, with a yaw root mean squared (RMS) error of 1 degree. Like the RTK GPS, this sensor is used for ground-truth purposes.

(a) Nautilus navigating on the ATLANTIS testbed.



(b) Droplets of water on the camera cage after. This is another barrier to the usage of visual sensors on ASVs.

Figure 3.3: Nautilus ASV.

Some other sensors are also available, as it is the case of a multibeam echosounder or a thermal camera. Those, however, are outside the scope of the present work. The sensors available to the SENSE ASV do not differ much from those present on Nautilus. Yet, SENSE was mainly used in the early stages of the work (see subsection 3.2.1). Both vehicles are built upon a Robot Operating System (ROS) [71] environment.

Sensors Calibration

The value of the data collected during campaigns in the ATLANTIS testbed cannot be dissociated from the quality of the measurements each used sensor does. Plus, as seen previously, classic VO approaches are dependent on accurate camera calibrations. This subsection covers the work carried to perform the calibrations of two of the sensors in use: the IMU and visual cameras.

The 9-DoF IMU in Nautilus outputs orientation, using sensor fusion to get the most out of each individual sensor. Regarding yaw or heading information, the sensor fuses data from a gyroscope and a magnetometer. Magnetometers make use of the Earth's magnetic field, which raises two problems. First, the magnetic field is neither constant along the Earth's surface nor constant over time. Second, materials surrounding an IMU can cause magnetic distortions, drastically affecting the magnetometer's measurements. The compensation of the deviation imposed by the Earth's magnetic field is done by adding the value of the magnetic declination on the location and date where data was collected. In Viana do Castelo, by the time data was being collected, magnetic declination (deviation of the Earth's magnetic north to the true north) measured at approximately -1.47 degrees. Concerning magnetic distortions caused by ferromagnetic objects near the IMU, a normalization procedure was performed on the measurements of the magnetic field strength.

For the stereo visual cameras, two distinct procedures had to be performed. First, each individual camera has to be calibrated, to represent the camera model. Then, a stereo calibration procedure

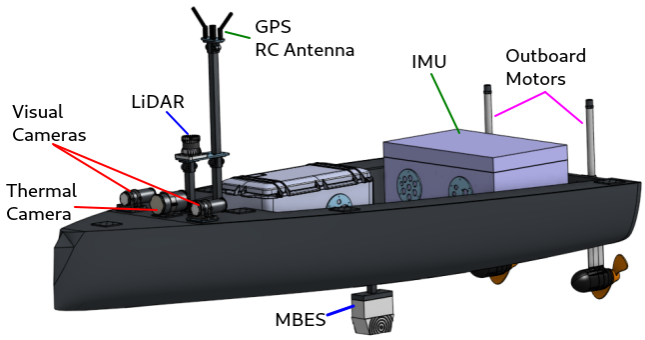


Figure 3.4: Nautilus sensors and actuators model.

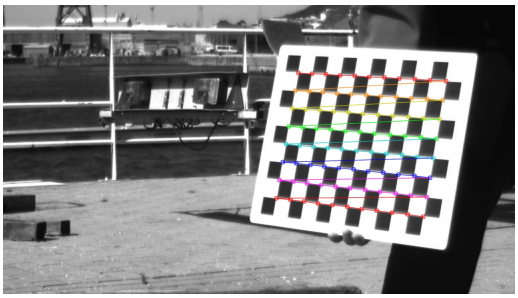
rectifies the image captured by both cameras, aligning epipolar lines (fig. 2.4) with the horizontal image axis. An OpenCV implementation[1] of the calibration technique proposed by Zhang [72] was followed. Intrinsic parameters are computed and then jointly optimized with lens distortions coefficients. After this calibration, 3D points are easily projected onto an image plane, and distortions are eliminated. Following an intrinsic calibration, a stereo rectification process was performed, meaning depth (or scale) information can be extracted based on the disparity of the same world point on images from the two cameras. Figure 3.5d depicts a point-cloud, with three-dimensional information computed from the stereo camera set.

3.2.1 Multi-layer Universal Architecture: sample application for data acquisition

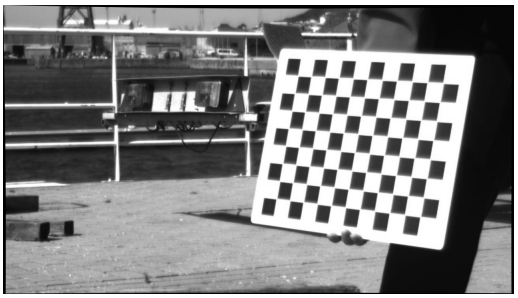
An autonomous navigation software architecture for data acquisition is proposed. It was used to define trajectories to be executed autonomously by the ASVs, based on an architecture that is easily re-configurable and reusable by different autonomous vehicles. A skill-based architecture is used, taking inspiration from the work by Pedersen et al. [73], expanding it and modifying the task level, so that the architecture can cope with complex O&M operations. With a focus on industrial robotics, skill-based architectures usually divide the architecture into several layers. The bottom layer, composed of actions or primitives, interconnects the hardware to the decision-making layer, where skills sit. This way, skills output data that is independent of the hardware, creating a level of abstraction that is much desired when software re-usability is wanted. On top of the skill level, a task level usually concatenates skills in a sequentially arranged fashion, allowing for distinct complex operations to be run one after the other.

Mobile robot applications of such architectures are, however, still very scarce. This autonomous navigation module aims to fill that void. Its main goal is to ease the programming task of mobile robots, and it tackles challenges such as software portability and re-usability, and data source agnosticism. In other words, it aspires to make the code interchangeable and independent from the protocol that drives data from sensors and to actuators. That said, this architecture has to fulfill the following requirements:

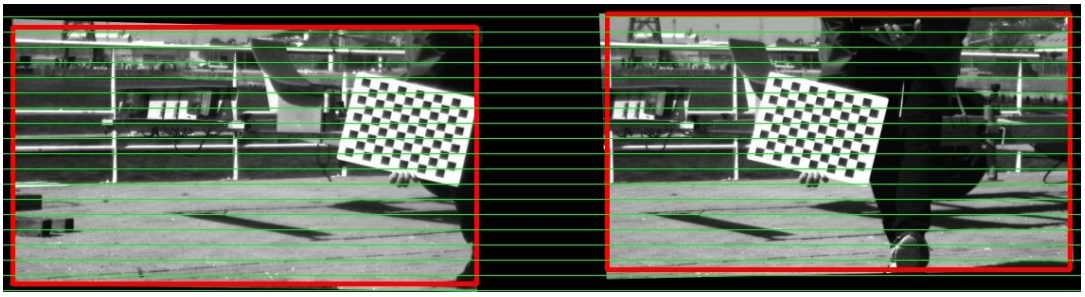
[1] Available at: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html (Accessed Jun. 22, 2023)



(a) One of the image frames used for the left camera's intrinsic calibration.

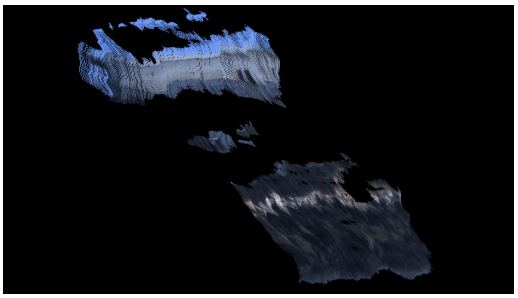


(b) Image frame 3.5a after calibration. Notice the appearance of dark borders.



(c) Stereo rectification. Green lines are the epipolar lines, made horizontal so that point search is constricted to one dimension.

(d) 2D image scene



(e) Example of a point-cloud (3D information) generated from the images of a stereo camera set. From the scene on figure 3.5c

Figure 3.5: Stereo camera set calibration.

- Each level of the hierarchy has to share information with the level atop or beneath it;
- A templated backbone for each level of the hierarchy must provide a mechanism that encompasses the basic operation of that level;
- Every level has to be agnostic to the levels within reach, meaning each component of the hierarchy might be swapped for a more (or less) complex component, without loss of the system's functionality.

Even though this architecture is similar to that proposed by Pedersen et al. [73], there are some differences worth pointing out. In the first one, one more layer is added, meaning this architecture is composed of four levels: a Mission, a Task, a Skill, and an Action level.

At the lowest level, actions are closely related to hardware. They are responsible for turning information into commands that the hardware level, through its drivers, can interpret. A skill used in certain domains and environments can only depend on specific actions that are physically compatible with the robot's hardware and with the operations to be performed.

The skills add functionalities to a robot and are defined as a sequence of individual and self-contained actions. Having some sort of control process, they interpret information from the environment, collected by sensors, and make decisions. Thus, skills can be seen as a sequence of actions. Skills can be either persistent, i.e. runs continuously in the background parallel to other skills until the task is stopped (e.g. odometry), or volatile, i.e. runs once until the goal condition is reached (e.g. positioning system). Programming at this level requires a comprehensive knowledge of the operation to be conducted. It is also required that the dynamics and limitations of the domain to which the skill is to be applied is taken into account. It is at this level that information is turned into commands to be fed into the action layer. Therefore, routines such as motion planning, motion execution, and obstacle avoidance should be implemented at this level.

At the task level, this architecture starts to differ from the already existing skill-based approaches. Unlike previous works, tasks are defined, not as a sequence of skills, but as a set of skills, which might be organised either sequentially, concurrently, or as a combination of both. Hence, tasks are a set of skills planned to overcome a specific goal. Tasks also define input parameters for skills. Unlike statically placed industrial robots, mobile robots might have to perform several skills concomitantly (e.g. hoovering while a manipulator performs an operation). With this approach, a robot can take advantage of concurrency, meaning it can benefit from skills that execute at the same time.

Above task level, it is proposed a mission layer, defined as a concatenation of tasks that are, again, run sequentially. Missions further extend the applicability of the proposed architecture, allowing for high-level programming of tasks, by simple definition of parameters. Upon the successful completion of all tasks, it can be guaranteed that a mission is also successfully concluded.

A visual representation of the architecture is depicted in Figure 3.6.

To better understand the division, one can picture the scenario in which an ASV has to inspect an offshore wind farm autonomously. The whole inspection operation composes the entire mission, which can be broken into smaller operations - tasks - such as the inspection of individual wind turbines. The inspection of each wind turbine requires the ASV to be able to perform navigation skills, such as reaching a location and hovering around a fixed point. Skills send velocity commands to actions, which convert that information to commands for the ASV motors.

Zooming in on each individual element of every layer (an individual mission, task, skill or action), one can find that the main workflow follows a similar sequence, which goes as follows:

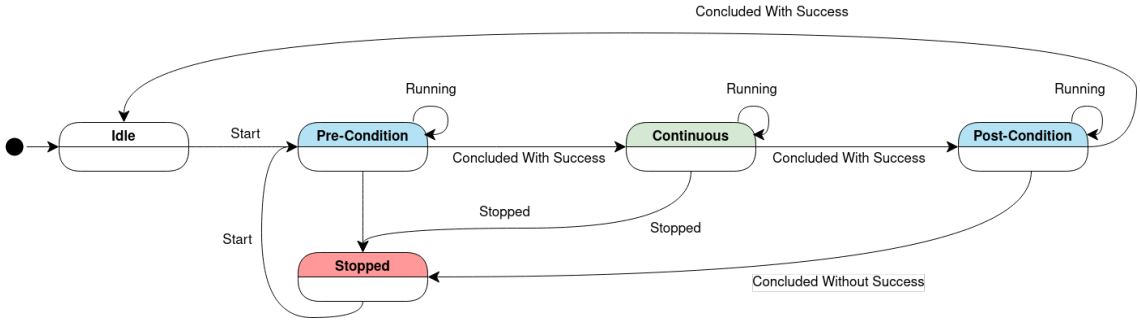


Figure 3.7: State machine diagram of the execution of each individual element of the proposed architecture.

1. Upon start, an initialisation routine is called. It is responsible for checking if the pre-conditions for the element's normal execution are met, and, if so, if it can proceed. Otherwise, the execution stops, returning an error;

2. The continuous execution routine is executed after the initialisation procedures are completed. This routine is non-blocking, meaning its execution is triggered by an object of a layer on top of it or, if it is the case of the highest layer, at a certain rate. Once again, in case of failure, the execution is halted, requiring a new start command to be sent;

3. Before concluding the execution, a post-condition checking routine is called. It is responsible for checking the overall execution status of the process and, eventually, for sending the last commands to the action layer.

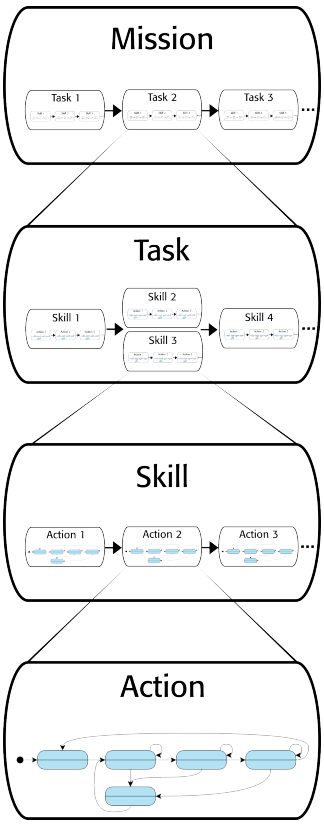


Figure 3.6: The four layers that make the proposed architecture.

In case of failure of any of the aforementioned routines, the execution enters a failure routine, which halts the operations. A visual representation of the described workflow is illustrated in Figure 3.7, in the form of a state machine diagram.

To illustrate the functionality of the proposed architecture, a task-level autonomous navigation algorithm was developed and deployed on both SENSE and Nautilus ASVs. A description of each level of the algorithm is described next.

The task takes an array of goal points of undefined size. Each element of that array is composed of three parameters, namely, x, y, and the point's reference frame. This means that coordinates can be expressed in the vehicle's initial odometry frame, or in the Universal Transverse Mercator (UTM) coordinate system. The task layer sends one goal-point at a time to the skill below and waits for its execution to be completed to send the next command. At the skill level, a positioning algorithm was developed. It receives a goal-point as a parameter, in the form of 2D coordinates, and

outputs linear and angular velocity commands to the action layer. Hence, position is represented by its x, y coordinates, and orientation, *yaw*, expressed by ψ. Data is driven by the ROS [71] messaging system and localisation information is published by a ROS implementation of an Extended Kalman Filter (EKF) [74], which fuses data from a GPS and from an IMU. The skill continuous execution state consists of a finite state machine model with five states, namely an initial state, a rotation phase, a state where the angular velocity is reduced, a cruising state, and a final state, where the ASV's velocity is decreased.

At the initial state, the ASV checks if its position relative to the goal point is within the limits set by a user-defined parameter, which bounds the maximum Euclidean distance - $\|\Delta(x,y)\|_2$ - from which the system advances directly to the final state. If $\|\Delta(x,y)\|_2$ is bigger than the defined parameter, the execution state changes to a rotation stage. At this stage, the ASV rotates until it is oriented to the direction given by a line that crosses both the vehicle's baseline and the goal-point. Given the dynamics of the environment where the ASV operates, and the lack of a finer velocity controller, the ASV stops the rotation when the module of its deviation in orientation - $|\Delta\psi|$ - is smaller than a user-defined parameter. Then, in the third state, the ASV waits for a time interval to pass, or until the error in orientation reduces in half, to start moving with linear velocity. The time interval duration is, once again, a parameter of the skill that a user can easily set or modify. In the following state, the ASV cruses at the maximum defined linear velocity. The angular velocity is calculated by a proportional controller, with a gain that was tuned during the conducted tests. Finally, as the ASV reaches the goal points' coordinates, and the skill's continuous execution enters its last stage, linear velocity is reduced to a specified value, and angular velocity is adjusted proportionally to $|\Delta\psi|$. The skill's continuous execution behaviour is illustrated in Figure 3.8.

3.2.2 Acquiring data from real-world environments

The results from the proposed architecture and the positioning skill implementation are presented and discussed, using two different sequences as examples. First, the discussion focuses on a smaller and easier succession of goal points, and then, a larger sequence, constrained to harsher environmental conditions, such as stronger undulation and wind, is also analysed. All data was acquired in the ATLANTIS Testbed. Sample code is publicly available [2]. For these experiments, the following parameters were defined:

- A parameter which bounds the maximum $\|\Delta(x,y)\|_2$ from which the system advances directly to the final state. It was set to 2.5 m;
- The maximum allowed deviation in orientation was set to 0.1 rad;
- The maximum allowed deviation in position was set to 1 m.

[2] https://github.com/edu-goncalves/MUSA

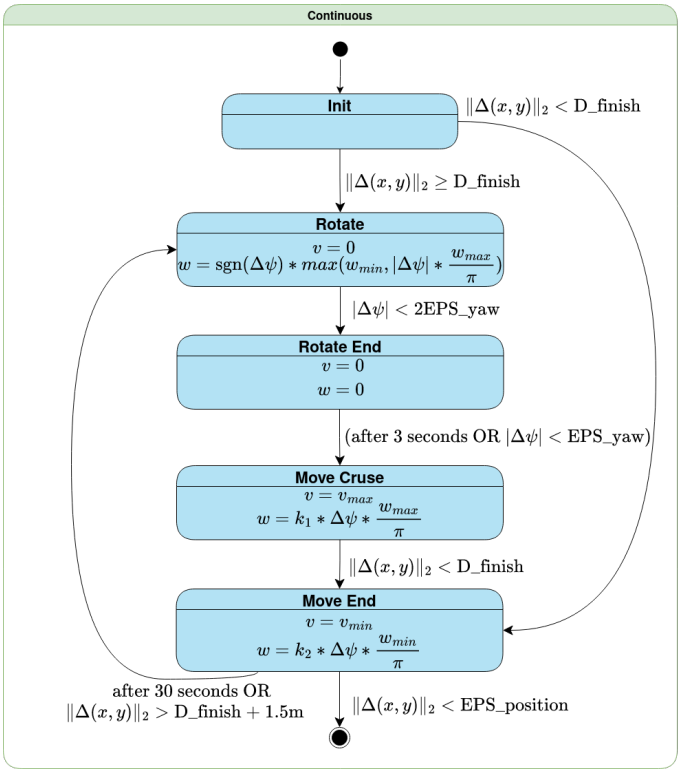
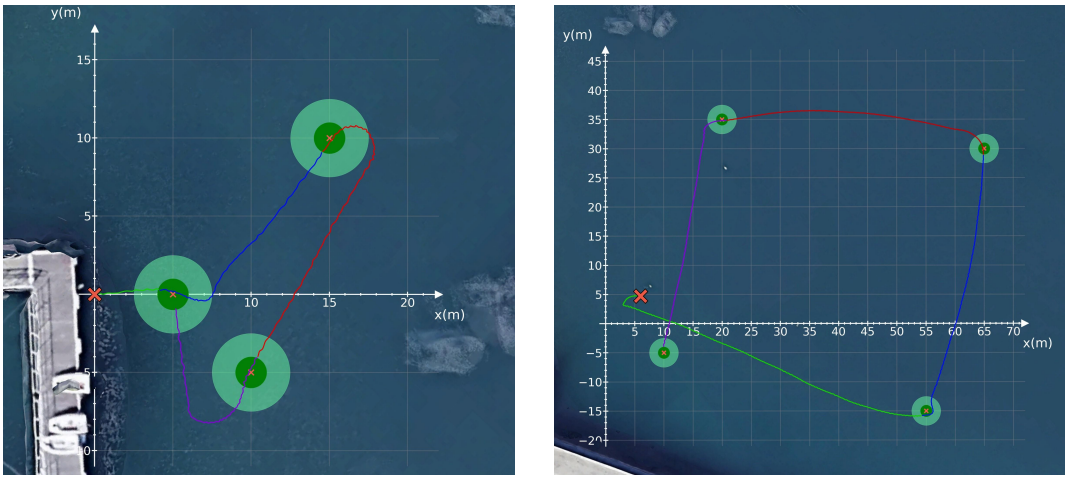


Figure 3.8: The state machine diagram of the continuous execution routine of the proposed positioning skill.

First, a local odometry frame was defined at the start of the operation. This frame was fixed to the geographic coordinates where the ASV was initialised, hence, the goal points were expressed in relation to the vehicle's starting pose, forming a set $P^O = \{p_1^O, ..., p_N^O\}$, with $p_n^O = (x_n^O, y_n^O), \forall n \in \{1, ..., N\}$, where '$O$' superscript represents the fixed odometry coordinate frame. Pose estimates were computed by fusing data from a RTK GPS and a 9 DoF IMU.

The first experiment consisted of a sequence of the goal points composed by the set $P^O = \{(5,0), (15,10), (10,-5), (5,0)\}$, and the path followed by the ASV is depicted in Figure 3.9a. The movement of the vehicle was constrained to a small area, forcing it to rapidly adjust to any deviations in orientation when heading to any of the goal points. The ASV was initially placed at coordinates $(0,0)$ on the odometry fixed frame, and started its trajectory oriented to the first goal-point $(5,0)$, meaning the execution started directly on the cruising state, at maximum linear velocity. Every point, except for the first one, forced the ASV to change its orientation, making sure that every stage of the continuous execution was performed. The ASV was able to successfully execute the task, reaching the 1 m tolerance defined at the beginning of the experiment.

A second experiment was conducted, simulating an inspection operation around the structure visible in figure 3.1. As it can be seen in Figure 3.9b, the scale of the movement was increased, easing constraints on the movement of the vehicle, but exposing it to harsher maritime conditions. The set of goal points was defined as $P^O = \{(55,-15), (65,30), (20,35), (10,-5)\}$, chosen so that the



(a) Task defined by the sequence of goal-points $P^O = \{(5,0),(15,10),(10,-5),(5,0)\}$

(b) Task defined by the sequence of goal-points $P^O = \{(55,-15),(65,30),(20,35),(10,-5)\}$

Figure 3.9: Results of task-level experiments performed in a port environment conducted in the ATLANTIS testbed. Light green circles represent the distance from the goal points from which the continuous execution state of the skill enters its last stage, while darker green circles represent the maximum allowed margin of error for the skill to succeed. Each line represents the execution of a single skill. The green line represents the first skill to be executed, followed by a blue, a red, and a purple line. Initial points are marked by a bigger red cross.

ASV trajectory could circumscribe the Durius floating structure (figure 3.10). At the beginning, and contrarily to the first experiment, the vehicle was placed facing the opposite direction to the first goal point, with coordinates $(6,5)$. As it rotated, the vehicle was being dragged by sea currents, as it can be observed in the first segment of the traced trajectory, represented by the green line in figure 3.9b. Despite the harsher conditions, the maximum acceptable margins of error were not changed. The task succeeded, meaning the ASV reached every goal point of the sequence with a maximum error lower than 1 m.

At action level, linear and velocity commands sent by the skill are converted into commands for the two electric thrusters, taking into account the kinematics of the ASVs.

After testing the autonomous navigation algorithm, a dataset was collected, consisting of data from a stereo set of visual cameras, a LiDAR, one IMU, and one GPS. For ground-truth purposes, an RTK-GPS and a 9-DoF IMU were used. Data was stored in a ROS message data format. The full dataset was collected over a full morning of navigation (more than an hour of navigation data), where the ASV navigated both in an autonomous and in a teleoperated fashion. The collected dataset is divided into seven different sequences. The first four sequences were collected by manually operating the ASV. Sequences 5 and 6 were collected autonomously, using the algorithm described in the previous section. Sequence 7 was collected as a mix of manual and autonomous operations. Sequences have a duration that ranges from 3 to 5 minutes, with the exception of

[3]SENSE performing an autonomous navigation task: https://youtu.be/GL83DWSML7s

Figure 3.10: SENSE performing an autonomous task, around the DURIUS platform [3].

sequence 4, with approximately 11 minutes of duration. Thus, data with the following properties were acquired:

- **Ground-truth signal** composed of an RTK-GPS, with an operating frequency of approximately 4Hz, and a 9-DoF IMU, providing measurements at 100Hz;
- **GNS/INSS** providing pose information at a rate of 4Hz;
- A **pointcloud** with a vertical and a horizontal resolution of 128 and 1024, respectively. Measurements were collected every 0.1 seconds;
- **Stereo visual images** from a synchronized pair of visual cameras, with HD resolution (1280x720p), at 10 fps.

3.3 Odometry Benchmark and Input Data

Even though visual/LiDAR-based odometry systems have been extensively studied in some environments, backed by autonomous driving and UAV-oriented datasets, there is still a lack of publicly available datasets, focused on visual/LiDAR data collected by vessels. That lack of publicly available datasets with a focus on visual/LiDAR data in maritime environments makes it very hard to assess the performance of pose estimation algorithms when exposed to environment characteristics such as undulation and very harsh atmospheric conditions. Hence, in this section, an evaluation of different odometry estimation methods is made, filling the gap that exists in the scientific community. The benchmark of such methods made it possible to come to conclusions regarding the performance of pose estimation algorithms in maritime conditions. Given that the used ASVs make use of the ROS [71], the conducted benchmark takes advantage of algorithms that run on such ROS environments. Moreover, purely single-modality odometry estimation systems were favoured over SLAM or absolute pose estimation implementations. This is because the

ultimate goal of this benchmark was to assess which data to be used by a DL fusion algorithm and maritime navigation does not usually provide the conditions to detect loop closures. Hence, six different methods were tested, and are listed below:

- Robot Localization (RL) [74]: a ROS implementation of nonlinear state estimators. Used to fuse data from the ASV's autonomous navigation module's GPS and IMU via an EKF. As an absolute localisation system (in what concerns position, at least), it does not suffer from position drifts, and position estimation errors are usually bounded by the performance of the sensors and disturbances created by the environment;

- ORB-SLAM [15]: the well-established V-SLAM implementation was tested. However, only the visual odometry thread was used, stripping the system of mapping and loop-closure modules. This was done for two reasons. First, when testing the algorithm's full implementation, it was clear that whenever the ability to track features was lost (which was very common), the system could very rarely recover, and pose estimation would not be computed. Second, no trajectory optimization is required, as the developed fusion module is to receive only relative pose transformations;

- RTAB-Map [75]: used for two distinct modalities. As RTAB-Map has modules for estimating odometry from different sensors, both a LiDAR-based (L) odometry and a stereo VO (V) implementation were tested. Both modalities rely on purely odometry systems. The VO module uses a 3D to 2D feature correspondence, comparing features from a current frame with those from a key frame. The LiDAR approach also compares current measurement with a key frame, calculating pose changes based on an ICP registration method, after applying a downsampling voxel grid filter;

- CT-ICP [76]: another ICP LiDAR odometry estimation method, applies a downsampling voxel grid filter to reduce the dimension of point-clouds, thus reducing the computational cost from ICP registration methods. Again, no loop closure is performed;

- Viso2 [77]: a VO library, also tested in a stereo mode. Produced awful estimations, regardless of the configuration parameters that were set. For that reason, its results are not taken into account in this benchmark.

Every odometry output is a 6DoF pose of the same fixed frame on the ASV. Ground-truth information has a earth-fixed referential, with position expressed in a UTM frame, and rotation relative to the earth's north pole.

For the evaluation of the trajectories estimated by each of the aforementioned methods, two distinct commonly used error metrics were calculated. As ASVs move on a 2D plane (assuming small variations on *roll*, *pitch*, and translations on z axis, imposed by undulation), evaluation is only performed on 3DoF (x, y, and *yaw*). The two used metrics are the root mean square (RMS) of the Absolute Trajectory Error (ATE) and the set of relative errors, ε, usually expressed as RE [78].

Given a ground-truth trajectory (aligned by the first pose estimate), the RMS of the ATE can be calculated as:

$$ATE_{pos} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}((x_i^{gt} - x_i^e)^2 + (y_i^{gt} - y_i^e)^2)}; \tag{3.6}$$

$$ATE_{\psi} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(\psi_i^{gt} - \psi_i^e)^2}, \tag{3.7}$$

where N is the number of evaluated poses, and the subscripts gt and e stand for ground-truth and estimated, respectively. The drawback of such a metric is that, as it computes the error through the whole extension of the trajectory, an error occurrence at the beginning of the trajectory is propagated through the subsequent pose estimations. Thus, RE is usually also used, as it is an evaluation metric on the KITTI dataset [36]. This metric iterates through segments of the trajectory with a pre-defined length (10%, 20%, 30%, 40%, and 50% of the total length of the trajectory), and evaluates them separately by aligning the first pose with the ground-truth, and measuring the error at the end of each sub-trajectory. Hence, and simplifying the metric used in [78] to a 2D evaluation problem, RE is the set of errors over all iterations, where the error at each iteration is defined as:

$$\varepsilon_{pos} = \sqrt{(\varepsilon_x)^2 + (\varepsilon_y)^2}; \tag{3.8}$$

$$\varepsilon_{\psi} = \psi_{gt} - \psi_e, \tag{3.9}$$

where ε_x and ε_y are the errors in x and y at the end of the sub-trajectory to be evaluated and are calculated as:

$$\varepsilon_x = x_{gt} - (\cos(\varepsilon_{\psi})x_e - \sin(\varepsilon_{\psi})y_e); \tag{3.10}$$

$$\varepsilon_y = y_{gt} - (\sin(\varepsilon_{\psi})x_e + \cos(\varepsilon_{\psi})y_e). \tag{3.11}$$

Usually, the average of RE is used, but other statistics, such as the median and the standard deviation, might be useful to assess the existence of outliers and, for that reason, failures on segments of the trajectory.

Every sequence of the dataset exposes the ASV to image features that are very far from the vessel, which can affect the quality of translation estimations made by VO systems. In the first three sequences, the ASV is not exposed to harsh undulation, and movement is constrained to a small area

(less than 1000m^2). Apart from sequence 2, every other sequence exposed the ASV to textureless image scenes, as depicted in figure 3.2. From sequences 5 to 7, as the ASV was subject to stronger undulation, water droplets on camera lenses harmed the quality of collected images. Table 3.1 sums up the obstacles present in each sequence of the dataset. The dataset has more than 8100 ground-truth measurements (RTK GPS and 9 DoF IMU) over the seven sequences.

Table 3.1: Obstacles for good odometry estimation by sequence.

	Sequence						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Fast Rotations	-	✓	✓	✓	-	-	✓
Distant Features	✓	✓	✓	✓	✓	✓	✓
Textureless Images	✓	-	✓	✓	✓	✓	✓
Autonomous Navigation	-	-	-	-	✓	✓	✓
Water on Lenses	-	-	-	-	✓	✓	✓
Undulation Intensity	+	+	+	++	++	++	++

Table 3.2: Absolute Trajectory Error and Relative Errors over seven different trajectories performed with the Nautilus ASV on the ATLANTIS coastal testbed.

Metric	**Method**	**Sequence**						
		1	**2**	**3**	**4**	**5**	**6**	**7**
ATE_{pos} (m)	CT-ICP	1.147	**0.542**	**0.668**	2.724	2.489	3.270	2.319
	ORB-SLAM	5.772	1.754	11.933	15.495	15.740	17.994	17.209
	RL	2.476	3.143	5.852	**0.701**	7.439	11.967	**0.839**
	RTAB-Map (L)	**1.061**	0.562	0.837	2.598	**2.042**	**2.877**	2.379
	RTAB-Map (V)	4.027	2.007	4.672	15.663	11.046	13.996	12.073
ATE_{yaw} (deg)	CT-ICP	2.938	1.116	**1.602**	8.376	0.741	0.386	0.592
	ORB-SLAM	18.196	1.568	60.666	18.461	50.972	74.521	76.876
	RL	3.308	**0.747**	18.796	**1.220**	7.218	3.617	2.482
	RTAB-Map (L)	**2.258**	1.059	1.754	5.672	**0.330**	**0.351**	**0.502**
	RTAB-Map (V)	12.437	9.015	28.386	41.979	53.577	49.266	48.373
RE_{pos} (%)	CT-ICP	4.590	3.479	3.967	4.947	8.959	10.676	7.547
	ORB-SLAM	34.584	11.440	46.842	36.406	47.190	66.503	50.799
	RL	9.248	22.052	21.852	**0.966**	31.634	12.097	**1.210**
	RTAB-Map (L)	**4.147**	**3.406**	**3.684**	4.823	**8.115**	**9.815**	7.639
	RTAB-Map (V)	23.614	18.052	26.053	23.997	38.417	45.567	36.680
RE_{yaw} (deg/m)	CT-ICP	0.062	0.101	0.080	0.095	0.020	0.017	0.013
	ORB-SLAM	0.389	0.120	2.178	0.461	1.053	1.368	0.927
	RL	0.114	**0.057**	1.137	**0.024**	0.070	0.086	0.019
	RTAB-Map (L)	**0.041**	0.095	**0.070**	0.091	**0.010**	**0.010**	**0.012**
	RTAB-Map (V)	0.407	0.585	0.743	0.615	0.729	0.934	0.656

Table 3.3: Ratio of failures per expected odometry estimation of VO systems.

Method	Sequence						
	1	**2**	**3**	**4**	**5**	**6**	**7**
ORB-SLAM	6.86%	0.18%	10.53%	5.75%	21.78%	41.79%	33.66%
RTAB-Map (V)	12.57%	15.18%	15.73%	9.54%	21.78%	29.46%	19.02%

In table 3.2, the results of each algorithm are presented. As methods that rely on LiDAR measurements have a superior performance on environments with a strong presence of planar surfaces, as is the case of the ATLANTIS coastal testbed, it comes as no surprise that LiDAR odometry algorithms achieved the best results. Furthermore, both RTAB-Map[75] and CT-ICP[76] use a point-to-plane approach to the ICP problem, further exploiting the planar scenario where data was acquired. On top of that, when collecting this dataset, not many dynamic objects disturbed the measurements acquired by the LiDAR (a rare occasion is depicted in figure 3.11), contributing to very good LiDAR odometry estimates. The performance of visual methods, on the other hand, confirmed the previously raised suspicions, given the texture-less and highly dynamic environment of the ATLANTIS testbed. However, one value from table 3.2 stands out. In sequence 2, the ORB-SLAM's [15] ATE_{yaw} was, unlike what is seen in other sequences, comparable to those obtained by non-visual methods. A closer inspection to the data leads to the conclusion that the second sequence was the only sequence where ORB-SLAM was able to run practically without failures (figure 3.13) from the beginning until the end. Table 3.3 showcases VO methods' percentage of failures per expected number of estimates over the seven sequences of the dataset. The rotation estimates made by the RL were greatly affected by the bad raw sensor data provided by the IMU.

Digging even further into the ATE metrics, it is noticeable that even though the RMS of the ATE is very large for VO methods, the median values of such error metric is consistently comparable to those of LiDAR methods, meaning that large deviations in the RMS of the ATE are due to failures of VO methods, and not because of the ineffectiveness of such methods to produce truthful estimations. The GNSS/INS localisation system produced mixed results, given the tendency of GNSS/INS to produce noisy measurements. Quantifying the RE results in an intra-modality fashion, RTABMap outperformed CT-ICP by 5.13% and 16.30% in translation and rotation, respectively. Regarding VO methods, RTABMap was, again, able to outperform the other algorithm under assessment, ORB-SLAM, by 26.05% in translation and 14.80% in rotation.

Even though these absolute and relative error metrics are widely used to assess the performance of localisation methods, they do not provide sufficient information about the value of each of the evaluated methods for a multi-modal odometry estimation system, the main focus of this dissertation. That multi-modal fusion system is to increase the robustness of single-modality-based systems, in the presence of harsh atmospheric conditions (wind and undulation), ferromagnetic structures and a highly dynamic scene. First, the odometry of each of these methods relates to

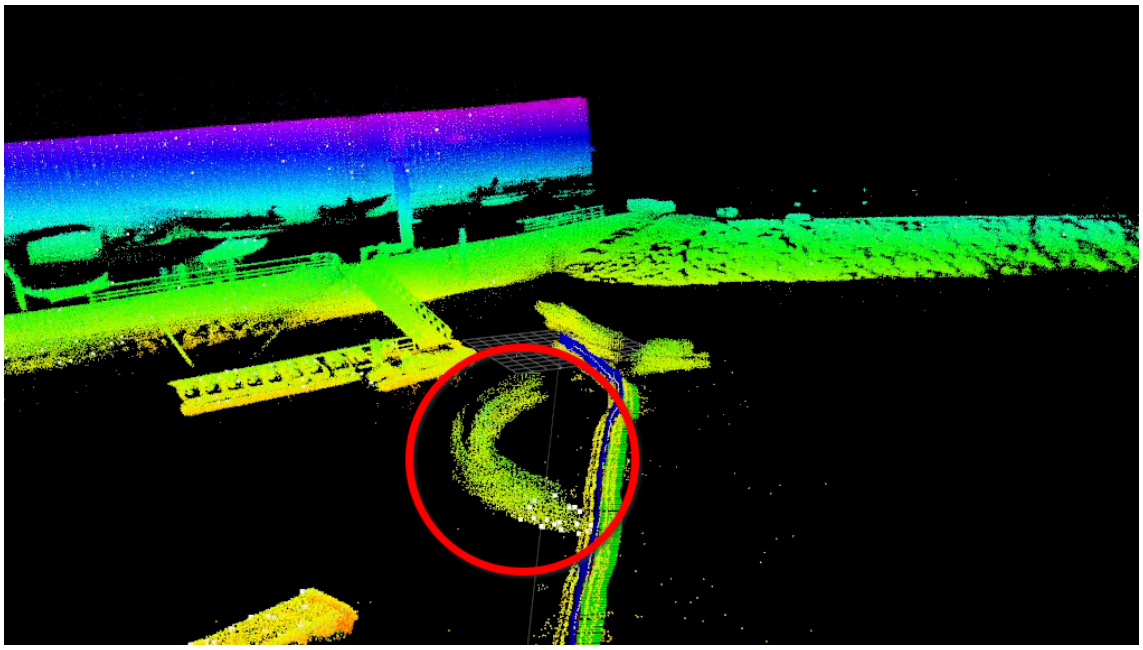


Figure 3.11: The influence (marked by a red circle) of a non-static craft in a 3D map generated with CT-ICP [76]. Such dynamic objects can disturb motion perception, affecting the performance of ICP cloud registration.

the initial pose of each method, which might, but most often is not, be shared among these odometry systems. Second, unsynchronized sensor measurements impose further challenges to fusion odometry methods. For that reason, the presence of spatio-temporal data misalignment has to be addressed, before feeding such information into a DL fusion module.

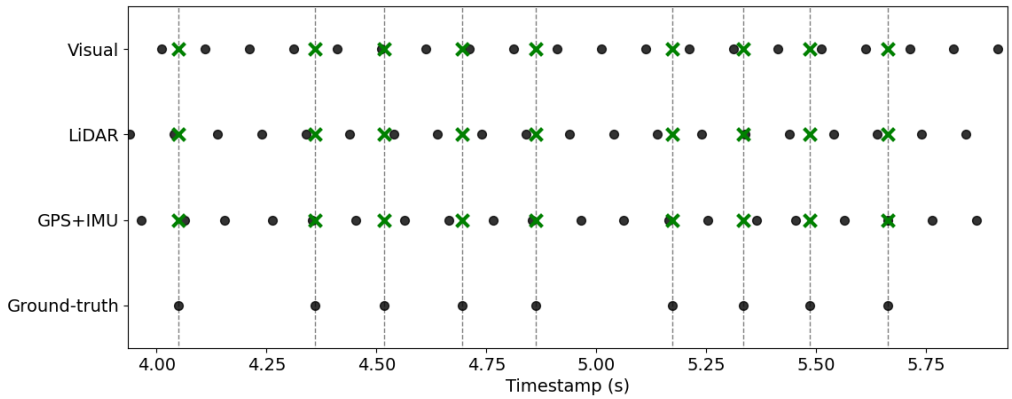


Figure 3.12: Sample of measurements from different sensors (black circles) and used pose estimations, after interpolation (marked by green crosses) to match the time-stamps of the ground-truth signal.

Addressing temporal misalignments first, an interpolation was performed, matching the time-stamps of the estimations of each odometry system with those of the ground-truth signal (fig. 3.12).

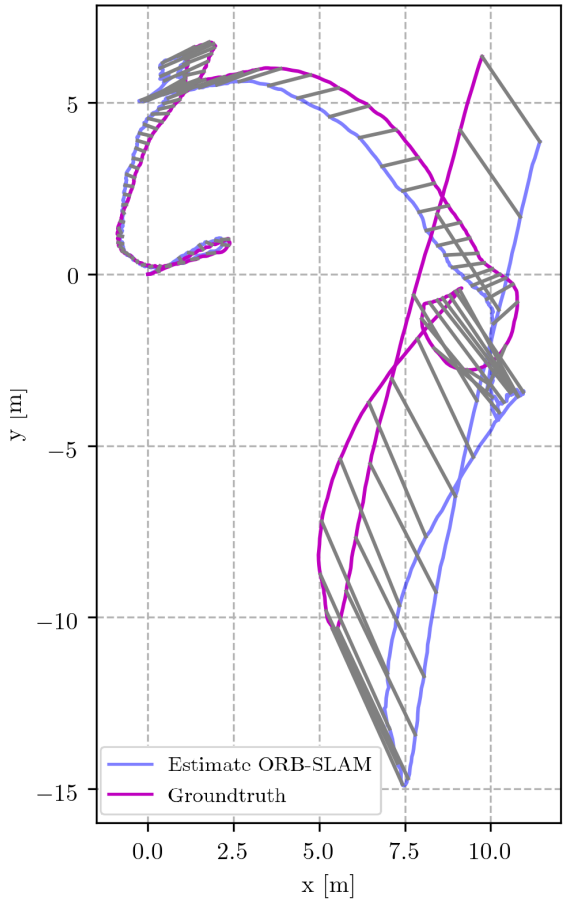


Figure 3.13: ORB-SLAM [15] trajectory estimation in sequence 2. Gray lines represent pose correspondences between the ground-truth and the estimated trajectory.

For rotation interpolation, represented by quaternions, spherical linear interpolation (SLERP) [79] was performed. Next, and having overcome the temporal misalignments of different measurements, the concepts introduced in section 3.1 are revisited, so that spatial misalignments can be circumvented. Instead of dealing with absolute poses, the fusion network is to receive relative poses as input, meaning each pose is relative to the previous pose estimated by each odometry estimation method. Let H be the set of n homogeneous transformations estimated by one of the previously mentioned systems, so that ${}_iH = \{{}^0H_1, {}^0H_2, ..., {}^0H_{n-1}, {}^0H_n\}$. To transform the k'th pose, ${}^0H_k$, to a relative pose, ${}^{k-1}H_k$, the following formula is applied:

$$ {}^{k-1}H_k = {}^{k-1}H_0 \cdot {}^0H_k = ({}^0H_{k-1})^{-1} \cdot {}^0H_k. \tag{3.12} $$

The representation of rotations is also changed, to avoid further complexity of the regression problem to be addressed, as quaternions and rotation matrices impose unit constraints that might be difficult to learn [80]. Thus, for orientation, an Euler Angles representation is chosen, as the dan-

ger of a gimbal lock occurrence is virtually nonexistent (pitch values are always very far from $\pm\frac{\pi}{2}$ radians), and not only are three independent values easier to visualize, but they also are simpler to regress.

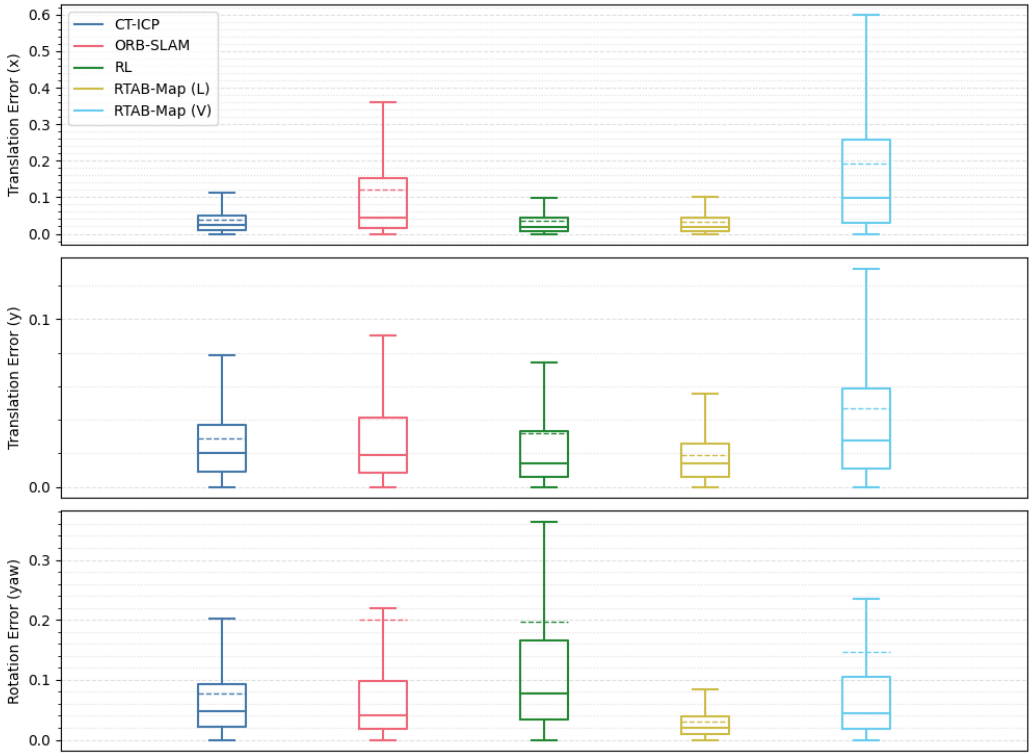


Figure 3.14: Boxplot representation of the absolute error of relative pose transformations. Translation errors in meters and rotation errors in degrees. For visualization purposes, outliers are not represented. The boxes extend from the first to the third quartile of the absolute error, with the median value in-between (solid horizontal line). Mean values are represented by a dashed horizontal line. The bottom and superior whiskers extend from the minimum error value to 1.5x the interquartile range.

The absolute errors of the relative poses for the full dataset are displayed in figure 3.14. A boxplot representation of the error was chosen, given that this simple representation provides a lot of information about the error dispersion of the methods under evaluation. Despite the change in used error metrics, LiDAR odometry estimation systems (especially RTAB-Map) proved to be the most reliable of all modalities, achieving the best performance, both in translation and rotation errors. The GNSS/INS system, however, produced solid translation estimates, but noisy orientation estimates were outperformed by every other method. Regarding VO systems, ORB-SLAM's performance surpassed that of RTAB-Map's VO module. In fact, the huge discrepancy between the median and mean values of VO systems' absolute errors is due to the lack of ability to extract reliable features, which halted or severely harmed localisation estimation computation, leading to error accumulation. If outliers were to be excluded from the analysis, however, one could easily

visualize that, when compared to CT-ICP, for instance, VO methods produced on-level rotation estimates, reinforcing the visual information's potential application in multi-modal fusion odometry systems. In fact, ORB-SLAM's median rotation error about the z axis was the second lowest of the five tested algorithms, surpassing RTABMap (V) by 5.12% and CT-ICP by 12.74%, only outperformed by RTABMap (L). Concerning translation errors, VO came short of achieving reasonable results, with a huge error dispersion. Error in translation in x saw an interquartile range (where half of the values of a boxplot graph sit) of 14 cm and 23 cm for ORB-SLAM and RTABMap (V), respectively. Those values contrast with the smaller dispersions of error of RTABMap (L) and RL, with 3.7 cm, and CT-ICP, with 4.0 cm. Finally, the median translation error in the x axis was the lowest for RL (1.7 cm), followed by RTABMap (L) (1.8 cm), CT-ICP (2.3 cm), ORB-SLAM (4.3 cm) and, coming last, RTABMap (V) (9.7 cm).

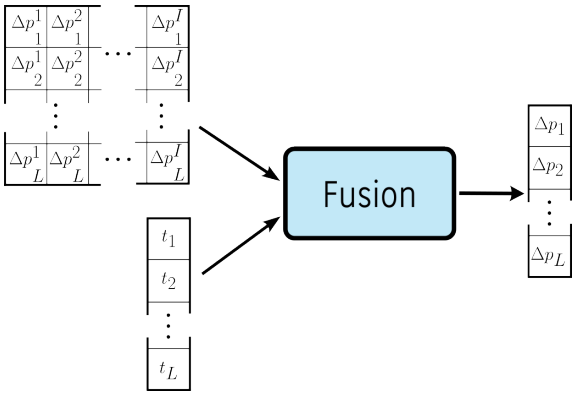
This section of the document confirmed the previously raised suspicions about VO algorithms and their inability to produce, on their own, truthful pose estimations in environments that severely affect image feature extraction. It showed, however, that even though maritime environments hinder the performance of such systems, suspending pose computations, some information can be extracted from these systems, especially rotation information. After this benchmark, three different modalities are chosen as inputs for the fusion module, presented in section 3.4, namely, ORB-SLAM [15], RL [74] and the LiDAR odometry module of RTAM-Map [75].

3.4 Self-Attention Fusion Module

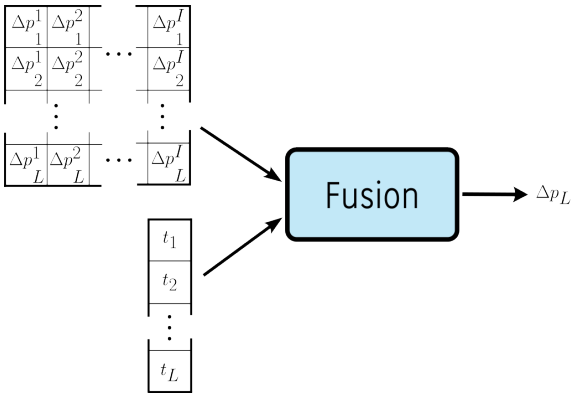
Transformer-based networks have been taking the AI world by storm. Recent NLP architectures, notably OpenAI's ChatGPT [4], exploit self-attention mechanisms to relate words within a sentence. Proposed by Vaswani et al., the original Transformer [43] forswore RNN models, as authors argued the self-attention mechanism would be able to learn spatial-dependencies of structured data. In this work, a self-attention-based model is chosen for the purpose of a late-fusion multi-modal odometry estimation system. It is done so that we can benefit from the positional encoding mechanism, given the irregularly spaced odometry measures over time. Hence, and as previously described in equation 3.4, the fusion system performs a multivariate regression task, to estimate a set of 6DoF poses, taking inspiration from the original Transformer architecture.

Let Δp_l^i be the l'th estimate of the i'th odometry estimation system, composed of an array with 6 elements (three translation values and three rotations). For an input sequence of length L, with I different systems, two fusion systems are proposed, depicted in figure 3.15. Also, let t_l be the l'th time-stamp of an array with length L. Two fusion systems are proposed. The first one (figure 3.15a) outputs an array of poses with length L, while the second (figure 3.15b) produces a single pose estimate, Δp_L.

[4] https://openai.com/blog/chatgpt



(a) Sequence to sequence fusion system's inputs and outputs.



(b) Sequence to sequence fusion system's inputs and outputs.

Figure 3.15: Graphical representation of the fusion systems' input and output data.

The original Transformer takes as an input, a sequence of word embeddings, meaning every word in a text has a learned vector representation, to get the most out of semantic relations of words. In this work, the proposed architecture benefits from that vectorial representation of data, where a set of three 6 DoF estimates are concatenated into a vector with size 18. This way, the interpolated estimates of three relative odometry estimation systems (described in the previous section) mimic the word embedding representation used by Vaswani et al. [43]. Unlike the original Transformer, here, the decoder module of the Transformer network is dropped, as the model relies only on the dependencies of the input signals, not on an input/output dependency.

Regarding the Positional Encoding, authors claimed that a sinusoidal representation of word positions would allow for different sequence lengths extrapolation, and proposed summing sinusoidal signals to every position along the text length, L, and varying the frequency of those signals along the word embedding dimension. Following that approach would raise the problem of losing temporal information about data and, for that reason, bigger temporal gaps between odometry estimates would be treated the same way as smaller ones. On the other hand, feeding estimations' timestamps into the positional encoding would lead to an irregular representation of the temporal

distribution of data which, when summed to the estimations themselves, would be very hard to learn. For that reason, instead of summing the positional information to the input signal, temporal information is concatenated to estimates, increasing the last dimension of the input data (even though Vaswani et al. [43] argued, in their work, that for the NLP task, concatenating positional information to the input produced similar results as summing that information as vectorial information). For that, two types of Positional Encodings were tested. First, a simpler representation of temporal information was tested, concatenating to each position of an input sequence a two-dimensional vector, $v \in \mathbb{R}^2$, of sinusoidal temporal information, such as:

$$v_1(t) = \sin(t * \beta); \tag{3.13}$$

$$v_2(t) = \cos(t * \beta), \tag{3.14}$$

where t is the timestamp of the estimates and β is a normalization constant to keep the domain of sine functions within the range of $[0, 2\pi[$. Then, and even though this temporal information representation considerably improved the ability of the network to produce odometry estimates, a learnable vector representation of time [81] was also tested, with $v \in \mathbb{R}^{18}$ being represented as:

$$v_i(t) = \begin{cases} \omega_i * t + \varphi_i & \text{if} \quad i = 1 \\ \sin(\omega_i * t + \varphi_i) & \text{if} \quad 2 \leq i \leq 18 \end{cases}, \tag{3.15}$$

where ω_i and φ_i are, respectively, learnable weights and biases. This vector representation of time further enhanced the model's predictions, as it is able to learn the periodicity of events, such as irregularly spaced timestamps.

In the original architecture, layer normalization [82] is performed after residual connection [83] operations (performed twice: after multi-head attention and after the point-wise feed-forward layer). During training, however, it was noticeable that the model had a hard time learning some patterns in data, especially when the temporal spacing between consecutive odometry estimations was bigger than usual (as depicted in picture 3.12). Performing layer normalization before the attention step and before the feed-forward layer boosted the quality of the predictions, as this prior normalization steps allows for faster convergence and makes the training phase less dependent on the optimizer's warm-up stage [84].

In the multi-head attention block, the following formula is applied:

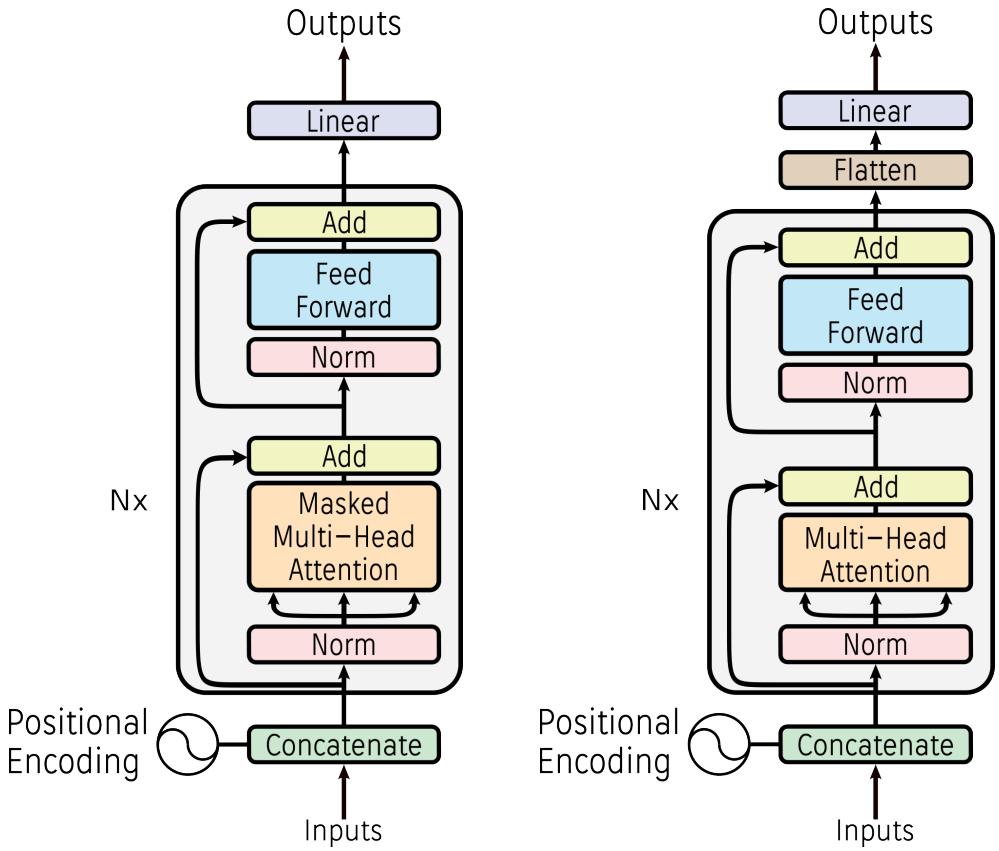
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M \right), \tag{3.16}$$

where M is a mask that doesn't allow a certain position to attend future estimates, Q, K and V, queries, keys, and values, respectively, are the learned linear projections of the output of the layer normalization step, and d_k is the dimension of the queries, keys, and values, in this case, 18 divided by the number of attention heads. Finally, at the end of the encoder, a linear layer projects a [Lx18] tensor into a [Lx6] tensor, meaning $L*6$ DoF poses.

Another model is also proposed, with two separate translation and rotation regressors. Still, the model follows the graphical representation depicted in 3.15a. For each regressor, the only change is in the last linear layer, where a [Lx3] tensor is estimated by each regressor, separately. Other architectures were implemented and tested, by flattening the output of the encoder and feeding it into a multilayer perceptron (MLP) or a simple linear layer (systems that replicate the graphical representation in figure 3.15b), producing only one odometry estimate at a time, and giving the same amount of information to every pose estimation. This caused the model to overfit training data and produce erratic estimates (see chapter 4). Overall, four different architectures are proposed. Architecture A, which has a single linear layer at the output of the transformer encoder. Architecture B and C flatten the regressor's output, feeding it to a linear or an MLP layer, respectively. Finally, architecture D, with separate translation and rotation regressors. Table 3.4 summarizes the tested architectures, and figure 3.16 graphically represents those models.

Table 3.4: Hyperparameters of the proposed Self-Attention fusion architectures. The names of the first three columns refer to the architecture that is appended to the output of the transformer encoder, while the last column refers to an architecture similar to the one to which a linear layer is appended, but that regresses either translation or rotation separately. L stands for the number of odometry estimates in a sequence.

Hyperparameters	**Architecture A**	**Architecture B**	**Architecture C**	**Architecture D**
Number of Inputs	$L \times 18$	$L \times 18$	$L \times 18$	$L \times 9$
Number of Outputs	$L \times 6$	6	6	$L \times 6$
Masked Attention	✓	✗	✗	✓
Number of Hidden Layers	-	-	3	-
Separate Regressors	✗	✗	✗	✓



(a) Linear layer at the output of the transformer encoder. With an input of length L, this architecture produces an output with the same length. This makes the full architecture A and is the backbone of architecture D, as it makes the individual translation/rotation estimates.

(b) The output of the encoder is flattened, and then, passed through a linear layer - architecture B - (or an MLP - architecture C). This architecture produces a single pose estimation at a time.

Figure 3.16: Proposed Odometry Fusion Networks. Nx indicates repetition of the Transformer encoder layer (gray box).

Chapter 4

Results of the self-attention based fusion system

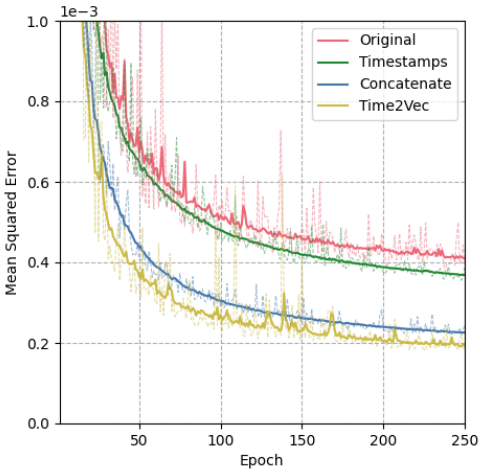
In this chapter, the results of the late fusion network proposed in section 3.4 (architecture D) are presented and discussed. The findings that justify the model's architecture and chosen hyper-parameters make the first section of the present chapter, and results over the seven sequences of the previously presented dataset are scrutinised.

In the first section of the chapter, the training results of different models are presented. Then, in section 4.2, the trajectories are reconstructed based on the relative poses estimated by the fusion algorithm, and the network's performance is compared to that of the single modality input methods. The models were trained in a modest machine, with an NVIDIA GeForce GTX 960M with 2 GB of dedicated memory, and 8 GB of RAM. Evaluation metrics are kept the same as the ones used in the previous chapter.

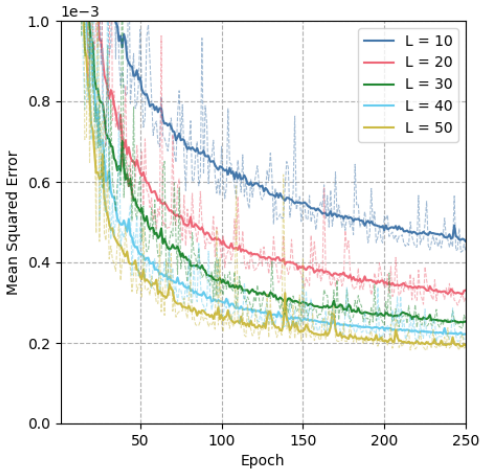
4.1 Model hyper-parameters and training

Section 3.4 presented several models for multi-modal odometry fusion, all based on the multi-head self-attention mechanism. For that reason, and following an identical approach to that used by Vaswani et al. [43], the AdamW optimiser [85] was used during training. Like the Adam optimiser [86], AdamW combines momentum with an exponential moving average to escape local minima. On top of that, AdamW adds weight decay regularisation to the algorithm, allowing for better generalisation of data. A varying learning rate was used, just like in [43], with a warm-up stage of 10 epochs. The mean squared error (MSE) of estimates was used as a loss function for all 6 DoF. Rotation regression could have used other loss functions (e.g. $\mathscr{L}(\theta, \theta_{gt}) = 1 - \cos(\theta_{gt} - \theta)$), but given that the normalised relative angles are always very small and, for that reason, very far from the limits of the normalisation interval ($[-180, 180[$ degrees), it was given preference for the simpler MSE loss over the more complex sinusoidal-based function.

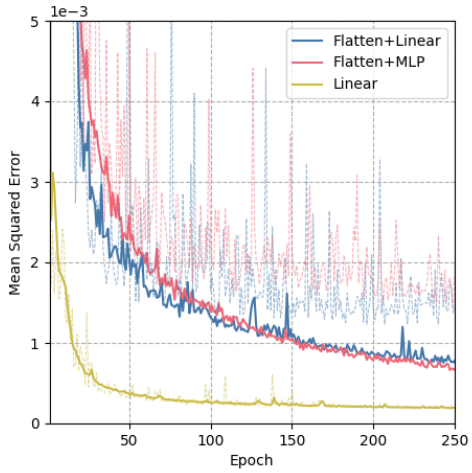
Given the small dimension of the available dataset, at least for the purpose of training DL models, seven different models were trained. Models share the same hyper-parameters but were trained with different data. For that matter, for the purpose of evaluating the performance of the fusion system in one sequence of the dataset, a model is trained with the remainder of the sequences, meaning that a model is always tested with data that the model was never exposed to. On top of that, additional data of moments when the ASV was in hovering conditions were also used, increasing training data with more than 9400 sets of multi-modal odometry estimates and further exposing the model to maritime environment scenarios and dynamics, adding to the 8100 sets of multi-modal odometry estimates that make the seven sequences of the previously presented dataset. The dimension of the testing data ranged from 4% to 15% of the total dataset. Training was done with a mini-batch of size 20. Training data was further split into a training (90%) and a validation set (10%).



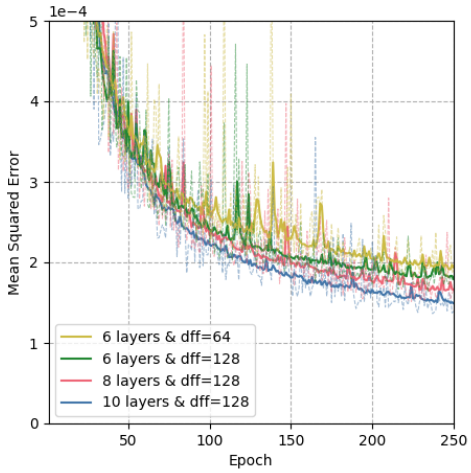
(a) Positional encoding.



(b) Sequence length.



(c) Pose-by-pose estimation over-fitting training data.



(d) Effects of increasing the architecture's dimensions.

Figure 4.1: Influence of hyper-parameters in training and validation loss (faded dashed lines). For reference, the losses of a common architecture (in yellow) are depicted in every chart.

In figure 4.1, the evolution of training and validation losses of the models presented in the previous chapter are shown. In sub-figure 4.1a, it is shown that summing a time-based positional encoding to the original signal yielded similar results to the originally used sine-based positional encoding. In fact, not much information is taken from this added positional encoding. There are two main reasons for that. First, and unlike the original Transformer where word embeddings are learned before the addition of positional information, here, the input is not learned but, instead, the raw pose estimates are fed into the model. Second, whereas in Vaswani et al. [43] the same signals are summed to a given word in a sentence, only depending on its position within the sentence, here, the signal being added is not constant, as instead of positions within a sequence, the message

time-stamp is used. This results in different vectorial representations of time for virtually every pose estimate, making it much harder to discern pose estimates from temporal information. For these reasons, concatenating temporal information to estimates leads to a significant decrease in training and validation losses. A Time2Vec [81] based positional encoding further improved the model's estimates. Overall, simply changing the data's positional information decreased training and validation losses by half, after 250 training epochs.

In sub-figure 4.1b it is shown that increasing the length of the sequences passed to the model leads to a decrease in the training loss. Five different sequence lengths were tested, and the results show that the more context it is given to the attention-based model, the better the results. It was chosen not to keep increasing the sequence length, as no significant improvements were seen, at the cost of heavier computations. In fact, a sequence of length 50 translates into a temporal window of more than 10 seconds, which provides a lot of information regarding the dynamics of an ASV.

Combining the information of a sequence of multi-modal odometry estimates into a single fused estimate via the flattening of the encoder's output tensor and feeding it into an MLP or a linear layer, which allowed for a single pose estimation, caused the models to overfit training data (sub-figure 4.1c shows that, in such models, the evolution of validation data was not able to keep up with that of training data).

Finally, the effects of increasing the model's capacity, by incrementing the number of encoder layers, or expanding the dimension of the feed-forward layer are depicted in sub-figure 4.1d. Further increasing the model's dimensions (more than 10 encoder layers or increasing the feed-forward's inner-layer dimensionality to more than 128) did not produce any noticeable changes. The scrutiny of the upcoming section focuses on a model that regresses position and orientation separately, with a Time2Vec positional encoding, 10 encoder layers, 4 attention heads, and a feed-forward with dimension 128.

4.2 Multi-modal odometry evaluation

In this section of the document, the model's performance assessment is done. That assessment starts with the model's output, hence, with relative poses, and then, the full reconstructed trajectories are presented, and compared against those estimated by the inputs of the network. These trajectory reconstructions make it possible to assess the model's ability to estimate an ASV absolute pose, and, for that reason, the appearance of errors and drifts in the estimates of a purely odometric system. Finally, an analysis of the behaviour of the fusion system when there are outages in the inputs of the network is also performed, drawing conclusions about the system's robustness to the constraints that a maritime environment imposes on sensors.

Starting this analysis with the relative pose estimates and, therefore, with the inputs and outputs of the fusion system, a significant improvement can be observed (figure 4.2). The fusion model

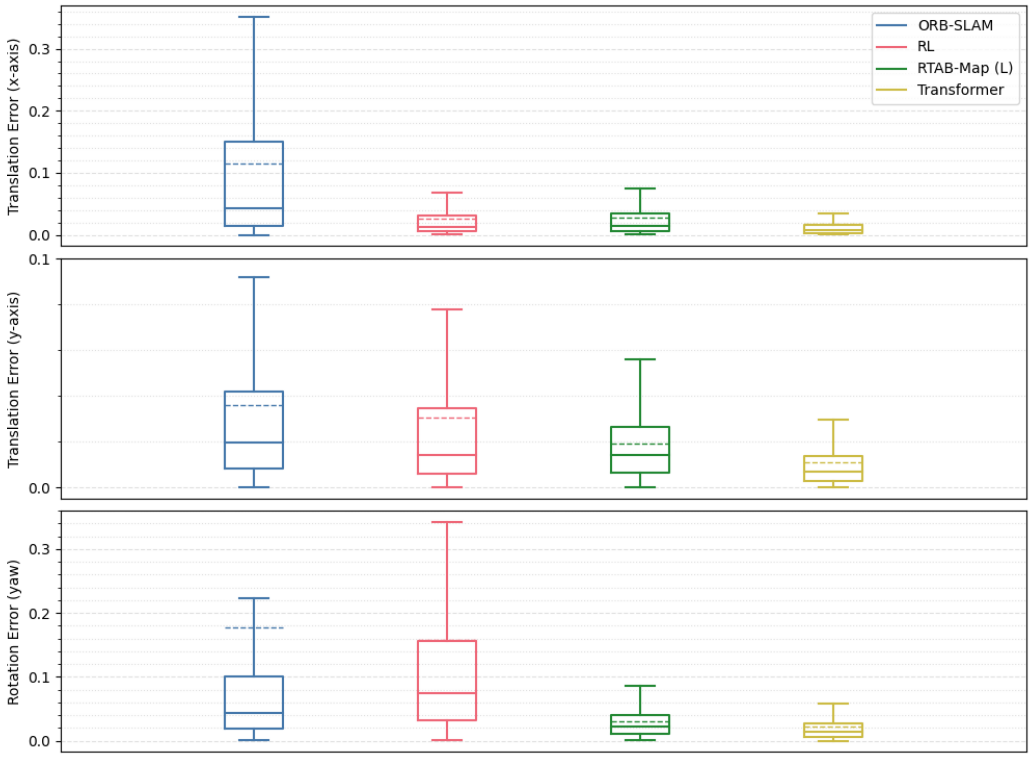


Figure 4.2: Fusion model absolute error of relative pose transformations against single-modality odometry systems. Translation error in meters and rotation error in degrees.

outperforms, by a large margin, the single-modality methods in the 3 DoF under evaluation. Translation estimates, especially in the direction of the x-axis, appear to have exploited the strengths of each modality. When compared to RL and RTABMap (L), the mean of translation error was decreased by 35.0% and 38.8%, respectively. The error's interquartile range and, for that reason, the error's dispersion also saw a meaningful reduction (51.2% and 55.5% in comparison with RL and RTABMap). Even though the errors of relative translations in the y-axis are not as impactful as those in the x-axis (the ASV's movement is constrained, mainly, to a translation in the x-axis and a rotation about the z-axis), they still have the potential to distort the trajectory to be reconstructed from the relative poses estimates, and should not be ignored as sea currents can drag the ASV, imposing lateral movement. The system's estimates concerning this DoF, improved mean estimates by more than 50% when compared to any of the inputs, and reduced the interquartile range by more than 45%. Finally, rotation about the z-axis also considerable improvements of more than 30% in every error metric.

As the ultimate goal is to access the quality of the reconstructed trajectories, the 6DoF estimated relative poses can lead to error accumulation, especially taking into account that the fusion network is a purely odometric system. For that reason, the trajectories of the seven sequences of the collected dataset were reconstructed (eq. 3.5). Table 4.1 summarises the relative error of the estimates over the full dataset.

Table 4.1: Relative Errors on the full Dataset.

Metric	Method	Error
RE_{pos} (%)	ORB-SLAM	38.451
	RL	11.319
	RTAB-Map (L)	5.579
	Fusion	**5.148**
RE_{yaw} (deg/m)	ORB-SLAM	0.793
	RL	0.135
	RTAB-Map (L)	0.058
	Fusion	**0.046**

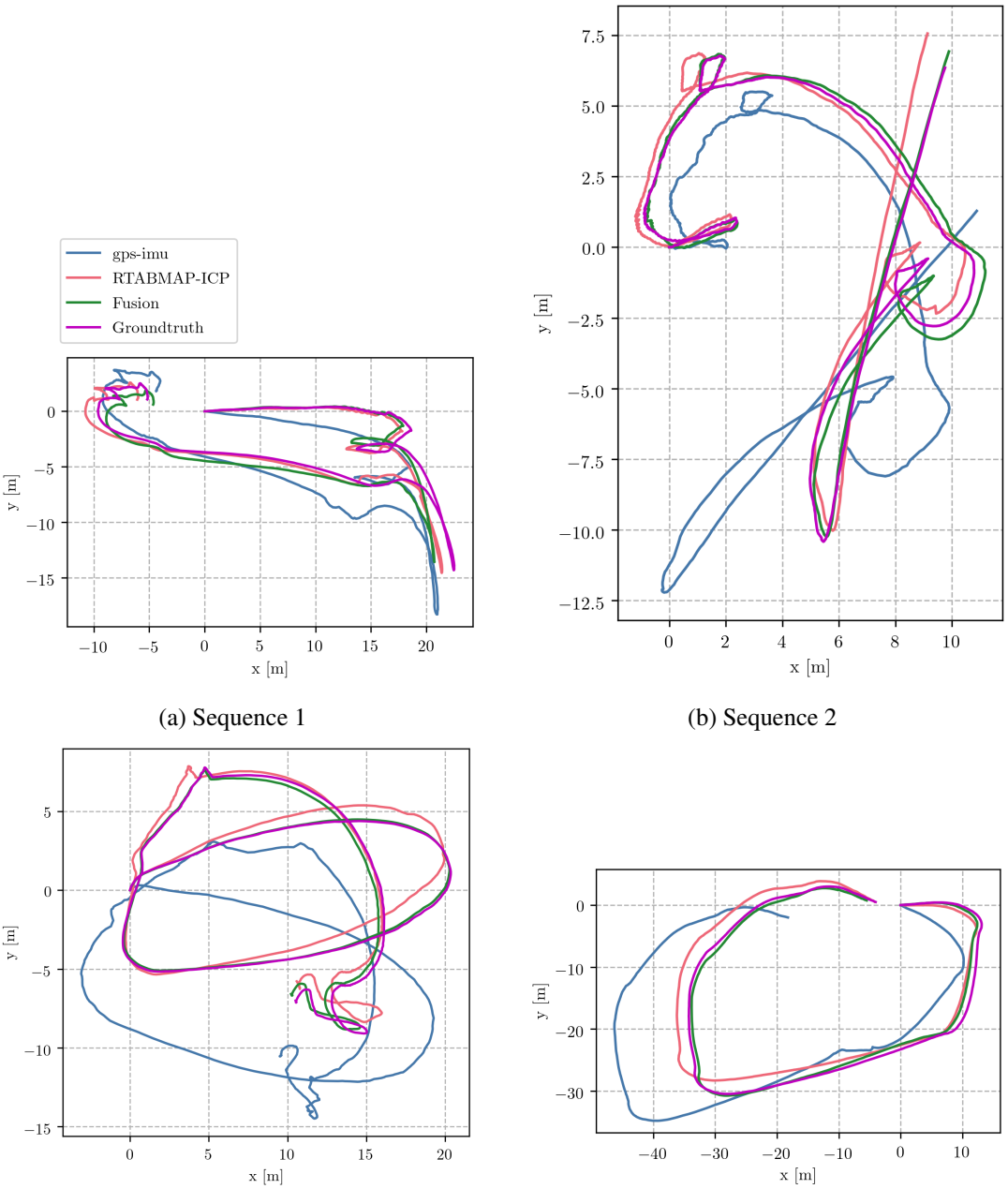
Even though there is still a significant improvement, both in translation and rotation metrics, the difference from the results of the fusion network to the LiDAR-based odometry system is less pronounced than it was when regarding only relative poses. Nevertheless, the multi-modal network outperformed RTABMap by 7.7% and 20.7% (translation and rotation), and GPS/INS system by 54.5% and 65.9% in translation and rotation, respectively. To better understand the results, the fusion system's performance is also evaluated in a similar fashion to what was done in section 3.3, dividing the dataset into the seven sequences that make it. Table 4.2 presents the ATE and RE achieved by each odometry estimation method, in each sequence. Again, trajectory errors (ATE) are based on a trajectory that is aligned by the first available common pose.

Table 4.2: Absolute Pose Odometry Errors with the Self-Attention Fusion model's results.

Metric	Method	Sequence						
		1	**2**	**3**	**4**	**5**	**6**	**7**
ATE_{pos} (m)	ORB-SLAM	5.772	1.754	11.933	15.495	15.740	17.994	17.209
	RL	2.476	3.143	5.852	**0.701**	7.439	11.967	**0.839**
	RTAB-Map (L)	1.061	0.562	0.837	2.598	2.042	2.877	2.379
	Fusion	**0.900**	**0.329**	**0.298**	3.930	**1.701**	**1.012**	2.968
ATE_{yaw} (deg)	ORB-SLAM	18.196	1.568	60.666	18.461	50.972	74.521	76.876
	RL	3.308	0.747	18.796	**1.220**	7.218	3.617	2.482
	RTAB-Map (L)	**2.258**	1.059	1.754	5.672	**0.330**	**0.351**	**0.502**
	Fusion	2.538	**0.690**	**0.975**	5.571	0.564	1.172	1.699
RE_{pos} (%)	ORB-SLAM	34.584	11.440	46.842	36.406	47.190	66.503	50.799
	RL	9.248	22.052	21.852	**0.966**	31.634	12.097	**1.210**
	RTAB-Map (L)	**4.147**	3.406	3.684	4.823	8.115	9.815	7.639
	Fusion	4.675	**3.022**	**1.405**	8.279	**3.969**	**0.827**	5.222
RE_{yaw} (deg/m)	ORB-SLAM	0.389	0.120	2.178	0.461	1.053	1.368	0.927
	RL	0.114	**0.057**	1.137	**0.024**	0.070	0.086	0.019
	RTAB-Map (L)	**0.041**	0.095	0.070	0.091	**0.010**	**0.010**	**0.012**
	Fusion	0.044	0.065	**0.026**	0.068	0.018	0.025	0.019

Delving into the data available in table 4.2, one can observe that the multi-modal fusion odometry system achieved the best trajectory position estimates in five of the seven sequences on the dataset. Given that the trajectories are reconstructed from the estimated relative poses, the position's ATE is a good indicator of the network's performance, as the reconstruction is dependent on all 6DoF

(position and rotation). In two sequences (sequences 4 and 7), however, the fusion system did not perform well when compared to single modality-based methods, nor when compared to the performance achieved in the remaining five sequences of the dataset. In fact, position errors, both ATE and RE, in these two sequences are discrepant with those achieved in the other sequences. A closer inspection of why that happens is done later in this section. Regarding orientation estimates, the results are, apparently, not as good as they are for position estimates. Actually, the RE of the rotation about the z-axis achieved the best results only in the third sequence. However, even though the fusion system was not able to consistently achieve the best rotation estimates when analysing the trajectories performed in every individual sequence, the estimates were consistently very good - the fusion network is always (with the exception of the third sequence) the second best method of the four under evaluation. This contributed to the overall performance presented in table 4.1. Rotation estimates were, therefore, very solid and unaffected by erratic or noisy inputs. Figure 4.3 displays the reconstructed trajectories of sequences 1, 2, 3 and 6. Sequences 4 and 7 are depicted in figure 4.4, and portray two situations where the fusion model was not able to achieve interesting results. ORB-SLAM's [15] results are not represented as in most sequences the estimates are too bad to visualize.



(a) Sequence 1

(b) Sequence 2

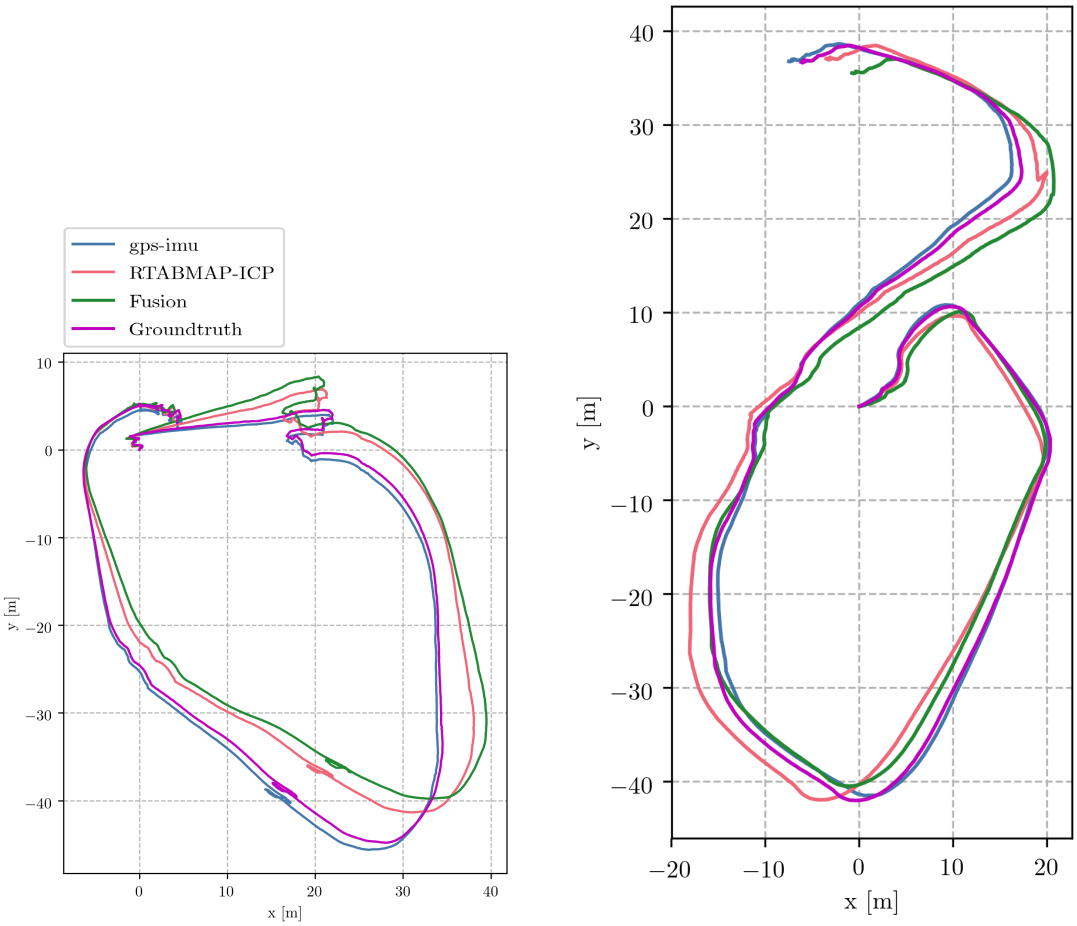
(c) Sequence 3

(d) Sequence 6

Figure 4.3: Trajectories estimates of the Fusion model, compared against its inputs.

If there is one thing that stands out from every sequence displayed in figure 4.3 is that the fusion model was able to ignore sudden leaps and distortions in the estimated trajectories. In the second sequence, for instance, RTABMap's estimate has an imperfection when the ASV crosses, approximately, the point with coordinates $(9\,\mathrm{m}, -2\,\mathrm{m})$. The fused odometry output was immune to that imperfection, leading to a smooth trajectory estimate. In the same sequence, the GPS/INS based localisation system did not close the loop in coordinates $(7\,\mathrm{m}, -5\,\mathrm{m})$, a flaw that is not noticeable in the fused trajectory. A similar pattern can be seen in sequence 3 (near the end of the RL es-

timated trajectory) and in sequence 6 (approximately halfway through the trajectory), distortions that the network promptly overcame.



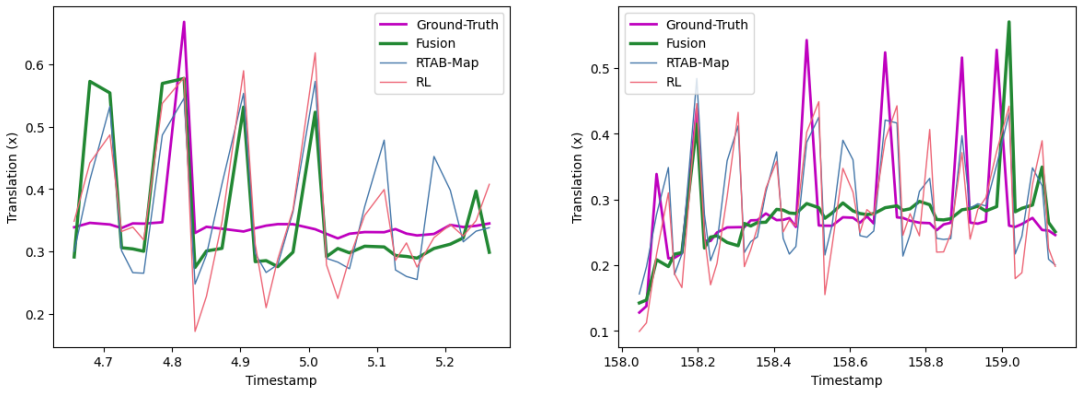
(a) Sequence 4

(b) Sequence 7

Figure 4.4: Results in sequences where the Fusion algorithm did not perform well.

In sequences 4 and 7, the network's ability to reduce the impact of distortions in the trajectories was very much alike what was already seen and discussed in the previous analysis (see RTABMap's trajectory, in sequence 7, at around coordinates $(20\,\text{m}, 25\,\text{m})$). Yet, the estimated trajectories are grossly defective, with no apparent reason for the inconsistent behaviour, especially taking into account the good performance achieved in the remaining sequences of the dataset. To understand what happened, figure 4.5 shows the relative translations in the x-axis. There is a significant presence of spikes in the data. It is noticeable that the spikes from the ground-truth signals and those that happen in the input signals are not coincident. The sharp variations in the ground-truth signal (cases where the signal assumes a value that is, approximately, double the signal values that are temporally close) represent a situation where the GPS signal was not successfully received, causing a bigger variation in the position estimates (the temporal interval between position estimates is also bigger). In these two sequences of the dataset, these GPS signal failures, however, were not

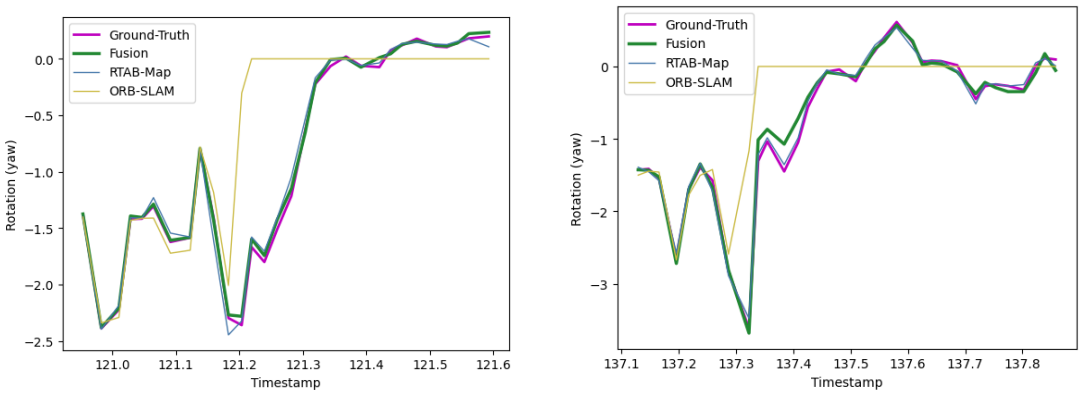
accompanied by a truthful time-stamp, affecting the interpolation step performed in the input data, which led to an unreal representation of temporal information and relative position estimates. The fusion algorithm was not able to capture these deficiencies, which led to the big deviations from the true trajectory, observed in figure 4.4.



(a) Sequence 4 (b) Sequence 7

Figure 4.5: Effects of erroneous time-stamps in GPS data in the multi-modal fusion system.

Section 3.3 exposed the frailty of VO systems in environments like the one made available by the ATLANTIS Coastal Testbed. It was shown that the high failure rate was responsible for the overall poor results over the full dataset, as in the second sequence, where ORB-SLAM [15] did not fail, the results were, even though not superb, acceptable. It was also shown that, in some sequences, the failure rate could go well beyond the 20% mark (table 3.3). The effects of those outages in the attention-based fusion model's results were, however, hardly noticed. In fact, the system was still able to perform smoothly in the sequences where the failure rate was higher (notably sequences 5 and 6). Regarding those sequences in particular, figure 4.6 shows the behaviour of the fusion network amid an outage of the VO system. It is noticeable that the output remained unscathed due to momentary input failures.



(a) Sequence 5 (b) Sequence 6

Figure 4.6: Multi-modal fusion system's behaviour with VO outages (rotation about the z-axis).

4.3 Discussion

In this chapter, the results of the fusion models described in section 3.4 of the previous chapter were presented and analysed. It was shown that increasing the length of the odometry sequence to be fed into the models highly impacted the training loss, as the model is given more or less context about the movement that is to be performed. It was also shown that replacing the positional encoding of the original Transformer [43] with a concatenated vectorial time-based encoding was enough to reduce training and validation error by half, allowing the model to cope with irregular time intervals between measurements. It was also shown that the proposed self-attention fusion models were able to outperform, in the collected dataset, the single modality odometry estimates of RTABMap's LiDAR-based system by 7.7% and 20.7% in position and rotation, respectively. When compared to the GPS/INS system, those values are raised to an improvement of more than 50%. These results were obtained when dealing with outages in the VO system, which produced no harm to the model's output.

Chapter 5

Conclusion and Future Work

The work conducted throughout the course of this dissertation culminated in an odometry fusion system based on DL techniques, that aimed at robustifying and improving the localisation estimates of an ASV. For that, a real-world dataset was collected, under harbor conditions, in a highly dynamic and harsh environment. Data consisted of GPS/INS information, LiDAR, and visual data. An autonomous navigation architecture is proposed, as it was used to perform some of the trajectories followed by the ASV. Then, a benchmark of different odometry estimation systems is done, to assess the strengths and flaws of each of the sensors that compose the dataset, in regard to the estimation of position and rotation information. Those odometry estimates are then used as the input of a robust odometry system, based on the self-attention mechanism and positional encoding techniques.

The collected dataset, and the odometry benchmark, tackle the lack of publicly available data with a focus on visual/LiDAR-based information, collected under maritime conditions. Even though such datasets have become extensively studied in some domains (autonomous driving and UAVs), there is still not sufficient available information about the performance of odometry estimation systems in harbor conditions. It was concluded that LiDAR-based odometry systems outperformed those that rely on visual and GPS/INS information by a great margin (50.7% and 57.0%, respectively). Even though VO methods performed poorly (failure rates, in some sequences of the dataset were higher than 30%), when taking into account only relative pose estimates, one VO method even outperformed a LiDAR-based odometry system by 12.74%, regarding orientation estimates. This justified the use of visual information as an input of a multi-modal fusion system, adding robustness to the estimates.

One of the challenges presented to the self-attention-based fusion system was how to incorporate temporal information from the odometry estimates into the input signal of the Transformer encoder. For that, it was shown that using a concatenated learned temporal-based positional encoding, reduced training and validation errors by half, after 250 training epochs. Relative pose estimates saw an overall mean improvement in translation of 35.0% and 38.8% when compared

to the GPS/INS and the LiDAR odometry method, respectively. Similarly, rotation about the z-axis was improved by more than 30%. Error dispersion was also greatly improved, as the error's interquartile range was reduced by over 50%. Turning relative pose estimates, and building the trajectory followed by the ASV, allowed for a finer assessment of the fusion system's performance. These reconstructed trajectories also saw significant improvements of 7.7% in translation and 20.7% in rotation over the full dataset.

Even though the fusion system proved to be resilient to failures in the VO inputs, there is still work to be done in order to refine the quality of the work that is described in this document. These improvements regard, primarily, the quantity and the quality of the data that was acquired in the ATLANTIS coastal testbed. Hence, the following improvements are yet to be performed:

- Correct the faulty time-stamps that, on occasion, can be detected in the RTK GPS signals;
- Improve the quality of visual information, as compressing the images to a Bayer filter applied to compress the images significantly degraded image quality;
- Increase the dataset, with three different purposes. First, the dataset was collected in a restricted environment, over the course of a single day. This means that the ASV navigated with undulation and wind conditions that did not differ much over the time when data was collected, not exposing the vessel to different weather conditions. Second, only visual faults are present in the dataset. It would be interesting to see faults in different modalities (such as GPS outages or LiDAR odometry impairments), and how the model would react to such flaws. Third, given the small dimensions of the dataset, overfitting of some of the proposed architectures was seen, which meant that the estimates produced by such models could not be accurately assessed.

References

[1] L. Kretschmann, H.-C. Burmeister, and C. Jahn, "Analyzing the economic benefit of unmanned autonomous ships: An exploratory cost-comparison between an autonomous and a conventional bulk carrier," *Research in Transportation Business and Management*, vol. 25, pp. 76–86, 2017.

[2] D. F. Campos, M. Pereira, A. Matos, and A. M. Pinto, "Diius - distributed perception for inspection of aquatic structures," 2021, in *OCEANS 2021: San Diego – Porto*, Conference Proceedings, pp. 1–5.

[3] A. M. Pinto, J. V. A. Marques, D. F. Campos, N. Abreu, A. Matos, M. Jussi, R. Berglund, J. Halme, P. Tikka, J. Formiga, C. Verrecchia, S. Langiano, C. Santos, S. N, J. J. Stoker, F. Calderoni, S. Govindaraj, A. But, L. Gale, D. Ribas, N. Hurtós, E. Vidal, P. Ridao, P. Chieslak, N. Palomeras, S. Barberis, and L. Aceto, "Atlantis - the atlantic testing platform for maritime robotics," 2021, in *OCEANS 2021: San Diego – Porto*, Conference Proceedings, pp. 1–5.

[4] M. I. Pereira, R. M. Claro, P. N. Leite, and A. M. Pinto, "Advancing autonomous surface vehicles: A 3d perception system for the recognition and assessment of docking-based structures," *IEEE Access*, vol. 9, pp. 53 030–53 045, 2021.

[5] D. F. Campos, A. Matos, and A. M. Pinto, "Modular multi-domain aware autonomous surface vehicle for inspection," *IEEE Access*, vol. 10, pp. 113 355–113 375, 2022.

[6] R. J. Silva, P. N. Leite, and A. M. Pinto, "Multi-agent optimization for offshore wind farm inspection using an improved population-based metaheuristic," 2020, in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Conference Proceedings, pp. 53–60.

[7] D. F. Campos, A. Matos, and A. M. Pinto, "An adaptive velocity obstacle avoidance algorithm for autonomous surface vehicles," 2019, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Conference Proceedings, pp. 8089–8096.

[8] M. I. Pereira, R. M. Claro, P. N. Leite, and A. M. Pinto, "Advancing autonomous surface vehicles: A 3d perception system for the recognition and assessment of docking-based structures," *IEEE Access*, vol. 9, pp. 53 030–53 045, 2021.

[9] R. Claro, R. Silva, and A. Pinto, "Detection and mapping of monopiles in offshore wind farms using autonomous surface vehicles," 2020, in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, Conference Proceedings, pp. 1–4.

[10] B. Shi, M. Wang, Y. Wang, Y. Bai, K. Lin, and F. Yang, "Effect analysis of gnss/ins processing strategy for sufficient utilization of urban environment observations," *Sensors (Basel, Switzerland)*, vol. 21, no. 2, p. 620, 2021.

[11] M. Bijelic, T. Gruber, and W. Ritter, "A benchmark for lidar sensors in fog: Is detection breaking down?" 2018, in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 760–767.

[12] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[13] A. M. Pinto, J. V. Amorim Marques, N. Abreu, D. F. Campos, M. Inês Pereira, E. Gonçalves, H. J. Campos, P. Pereira, F. Neves, A. Matos, S. Govindaraj, and L. Durand, "Atlantis coastal testbed: A near-real playground for the testing and validation of robotics for o&m," 2023, in *OCEANS 2023 - Limerick*, pp. 1–5.

[14] D. Scaramuzza and F. Fraundorfer, "Tutorial: Visual odometry," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[15] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[16] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," 2013, in *2013 IEEE International Conference on Computer Vision*, pp. 1449–1456.

[17] A. Singh, "An opencv based implementation of monocular visual odometry," Indian Institute of Technology Kanpur, Report, 2015.

[18] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin Heidelberg, 2006, in *Computer Vision – ECCV 2006*, Conference Proceedings, pp. 430–443.

[19] C. Tomasi and T. Kanade, "Detection and tracking of point features," International Journal of Computer Vision, Tech. Rep., 1991.

[20] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," 2010, in *2010 IEEE Intelligent Vehicles Symposium*, Conference Proceedings, pp. 486–492.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981, cited By :16117 Export Date: 30 December 2021.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[23] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," 2011, in *2011 International Conference on Computer Vision*, Conference Proceedings, pp. 2564–2571.

[25] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[26] R. Wang, M. Schwörer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," Venice, Italy, October 2017, in *International Conference on Computer Vision (ICCV)*.

[27] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," September 2014, in *European Conference on Computer Vision (ECCV)*.

[28] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," 2015, in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2938–2946.

[29] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," 2017, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6555–6564.

[30] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization," 2017, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2652–2660.

[31] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," 2017, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050.

[32] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," 2015, in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766.

[33] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, 2017.

[34] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," 2018, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6939–6946.

[35] Y. Lin, Z. Liu, J. Huang, C. Wang, G. Du, J. Bai, and S. Lian, "Deep global-relative networks for end-to-end 6-dof visual localization and odometry," A. C. Nayak and A. Sharma, Eds. Cham: Springer International Publishing, 2019, in *PRICAI 2019: Trends in Artificial Intelligence*, pp. 454–467.

[36] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," 2012, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Conference Proceedings, pp. 3354–3361.

[37] Y. Shavit and R. Ferens, "Introduction to Camera Pose Estimation with Deep Learning," *arXiv e-prints*, p. arXiv:1907.05272, Jul. 2019.

[38] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "DeMoN: Depth and motion network for learning monocular stereo," 2017, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5622–5631.

[39] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," 2017, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619.

[40] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," 2018, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1983–1992.

[41] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," 2018, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7286–7291.

[42] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera," 2019, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7062–7071.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[44] X. Li, Y. Hou, P. Wang, Z. Gao, M. Xu, and W. Li, "Transformer guided geometry model for flow-based unsupervised visual odometry," *Neural Computing and Applications*, vol. 33, no. 13, pp. 8031–8042, Jul 2021. [Online]. Available: https://doi.org/10.1007/s00521-020-05545-8

[45] L. R. Agostinho, N. M. Ricardo, M. I. Pereira, A. Hiolle, and A. M. Pinto, "A practical survey on visual odometry for autonomous driving in challenging scenarios and conditions," *IEEE Access*, vol. 10, pp. 72 182–72 205, 2022.

[46] Ø. Volden, A. Stahl, and T. I. Fossen, "Vision-based positioning system for auto-docking of unmanned surface vehicles (usvs)," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 1, pp. 86–103, 2022.

[47] T. Kriechbaumer, K. Blackburn, T. P. Breckon, O. Hamilton, and M. Rivas Casado, "Quantitative evaluation of stereo visual odometry for autonomous vessel localisation in inland waterway sensing applications," *Sensors*, vol. 15, no. 12, 2015.

[48] S. Wang, Y. Zhang, and F. Zhu, "Monocular visual slam algorithm for autonomous vessel sailing in harbor area," 2018, in *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, Conference Proceedings, pp. 1–7.

[49] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[50] Y. Ren and X. Ke, "Particle filter data fusion enhancements for mems-imu/gps," *Intelligent Information Management*, vol. 2, pp. 417–421, 2010.

[51] J. Vasconcelos, C. Silvestre, and P. Oliveira, "Ins/gps aided by frequency contents of vector observations with application to autonomous surface crafts," *Oceanic Engineering, IEEE Journal of*, vol. 36, pp. 347–363, 2011.

[52] W. Naeem, R. Sutton, and T. Xu, "An integrated multi-sensor data fusion algorithm and autopilot implementation in an uninhabited surface craft," *Ocean Engineering*, vol. 39, pp. 43–52, 2012.

[53] M. Romanovas, R. Ziebold, and L. Lança, "A method for imu/gnss/doppler velocity log integration in marine applications," 2015, in *2015 International Association of Institutes of Navigation World Congress (IAIN)*, pp. 1–8.

[54] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," 2015, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Conference Proceedings, pp. 298–304.

[55] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[56] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," 2015, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Conference Proceedings, pp. 2174–2181.

[57] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," 2021, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5692–5698.

[58] J. C. Leedekerken, M. F. Fallon, and J. J. Leonard, *Mapping Complex Marine Environments with Autonomous Surface Craft*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 525–539.

[59] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," 2017, in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, Conference Proceedings, pp. 3995–4001.

[60] B. Teixeira, H. Silva, A. Matos, and E. Silva, "Deep learning for underwater visual odometry estimation," *IEEE Access*, vol. 8, pp. 44 687–44 701, 2020.

[61] Y. Almalioglu, M. Turan, M. R. U. Saputra, P. P. B. de Gusmão, A. Markham, and N. Trigoni, "SelfVIO: Self-supervised deep monocular visual–inertial odometry and depth estimation," *Neural Networks*, vol. 150, pp. 119–136, 2022.

[62] Y. Wan, Q. Zhao, C. Guo, C. Xu, and L. Fang, "Multi-sensor fusion self-supervised deep odometry and depth estimation," *Remote Sensing*, vol. 14, no. 5, 2022.

[63] B. Li, M. Hu, S. Wang, L. Wang, and X. Gong, "Self-supervised visual-lidar odometry with flip consistency," 2021, in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3843–3851.

[64] A. Javanmard-Gh, D. Iwaszczuk, and S. Roth, "Deeplio: Deep lidar inertial sensor fusion for odometry estimation," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 51, pp. 47–54, 2021.

[65] N. Kaygusuz, O. Mendez, and R. Bowden, "Multi-camera sensor fusion for visual odometry using deep uncertainty estimation," 2021, in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2944–2949.

[66] C. Bishop, "Mixture density networks," Aston University, WorkingPaper, 1994.

[67] S. Pillai and J. J. Leonard, "Towards visual ego-motion learning in robots," 2017, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5533–5540.

[68] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," September 2018, in *Proceedings of the European Conference on Computer Vision (ECCV)*.

[69] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," Jul. 2016, in *arXiv:1607.02565*.

[70] J. Diebel, "Representing attitude : Euler angles , unit quaternions , and rotation vectors," 2006.

[71] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009, in *ICRA Workshop on Open Source Software*.

[72] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[73] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. S., S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[74] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system." Springer, July 2014, in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*.

[75] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[76] P. Dellenbach, J. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: real-time elastic lidar odometry with loop closure," *CoRR*, vol. abs/2109.12979, 2021. [Online]. Available: https://arxiv.org/abs/2109.12979

[77] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," 2011, in *Intelligent Vehicles Symposium (IV)*.

[78] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," 2018, in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*.

[79] K. Shoemake, "Animating rotation with quaternion curves," ser. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, 1985, in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, p. 245–254. [Online]. Available: https://doi.org/10.1145/325334.325242

[80] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," *arXiv e-prints*, p. arXiv:1709.08429, 2017.

[81] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, "Time2vec: Learning a vector representation of time," 2019.

[82] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv e-prints*, p. arXiv:1607.06450, Jul. 2016.

[83] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

[84] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, "On layer normalization in the transformer architecture," 2020.

[85] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv e-prints*, p. arXiv:1711.05101, Nov. 2017.

[86] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.