FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deep Learning Approaches Assessment for Underwater Scene Understanding and Egomotion Estimation

Bernardo Teixeira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Doutor Eduardo Silva
Co-Supervisor: Prof. Doutor Anibal Matos

September 13, 2019

© Bernardo Teixeira, 2019

Resumo

A progressiva utilização de soluções baseadas em visão computacional no âmbito do desenvolvimento de veículos autónomos submarinos (VAS) vem sendo impulsionada pelo baixo preço das câmaras visiveis em comparação com outros tipos de sensorização, tendo em conta a quantidade de informação que pode ser retirada apartir da análise da informação visual. No contexto de ambientes subaquáticos, em que soluções como o GPS não são viáveis e outros tipos de sensorização já mostraram vulnerabilidades no passado, a utilização de câmaras e novos algoritmos que utilizam a informação obtida por estas, tem sido alvo de imensa pesquisa e desenvolvimento.

No entanto, e não obstante a maturidade e os iterativos ganhos em performance dos algoritmos de aplicação de visão computacional, a sua dependência de formulações geométricas elaboradas, combinada com alguma incapacidade de manter elevados níveis de performance em cenários de aplicação mais complexos, comprometem a sua aplicação em soluções robóticas em meio operacional. O elevado volume de dados de sensorização produzido pelas soluções robóticas atuais, aliado ao maior poder de processamento, realça a possibilidade de implementação de métodos computacionais baseados em aprendizagem em soluções de estimação de movimento e navegação robótica.

Nos últimos anos, abordagens baseadas em arquitecturas Deep Learning tem aparecido como métodos capazes de solucionar problemas de visão computacional. A performance dessas abordagens rapidamente ultrapassaram os resultados obtidos pelos métodos clássicos em alguns tipos de tarefas no âmbito da visão computacional, nomeadamente na detecção e classificação de objectos. No entanto, o seu potencial de aplicação não se cinge a estas tarefas, existindo trabalho anterior também em aplicações de navegação robótica.

Nesta dissertação, o foco é colocado na avaliação da performance dos métodos de Deep Learning em tarefas de relocalização e estimação de movimento, em particular transportando os métodos do estado da arte da literatura para o contexto subaquático, testando e avaliando a sua performance num dataset visual subaquático desenvolvido no decorrer deste trabalho, que contém cenários de aplicação reais de uma solução robótica em missão operacional.

Baseado nos resultados obtidos, foram desenvolvidos duas novas arquitecturas de fusão sensorial para optimização do posicionamento global do robô com o propósito de corrigir o *drift* acumulado pelos métodos de estimação de movimento; uma rede de fusão de informação visual e inercial baseada num esquema de aprendizagem supervisionado e uma rede neuronal recorrente que realiza a fusão de estimativas do posicionamento provenientes de múltiplas cameras que não possuem *overlap* dos seus respectivos campos de visão.

Abstract

The widespread use of Computer Vision approaches on Autonomous Underwater Vehicles (AUV) applications, rises from the fact that vision sensors are low cost and can provide lots of information for a variety of tasks the AUV has to perform. In the underwater context a lot of research has been devoted to the development of algorithms that leverage visual camera information.

However, and despite the technical maturity and accurate performance of visual based algorithms, classical visual approaches rely on intricate geometric formulations that may struggle to capture more complex environments. In addition, robotic solutions of today provide huge amounts of sensor data, begging the question of whether more data-centric approaches could be beneficial for improving accuracy and performance of visual-based tasks.

Deep learning approaches for Computer Vision applications have been surfacing in the last couple years, posing as a viable alternative to classical methods for a wide range of computer vision tasks, even severely outperforming classical algorithms for some visual tasks. Despite of its prevalence in computer vision tasks such as object detection and scene classification, the realm of potential applications can still be further extended. End-to-end visual-based navigation applications, tailored for several different tasks ranging from scene localization to egomotion estimation, could result in interesting alternative approaches to the robot navigation problem.

State-of-the-art algorithms have shown the tremendous potential deep learning architectures can have for visual navigation implementations, though they are still mostly outperformed by classical feature-based techniques.

In this work, we apply the advancements in deep learning methods for visual-based robot navigation to the more challenging underwater environment, providing both an underwater visual dataset acquired in real operational mission scenarios and an assessment of state-of-the-art algorithms on the underwater context.

Furthermore, we propose two novel pose optimization architectures for the purpose of correcting visual odometry estimate drift: a Visual-Inertial fusion network aiming to correct monocular estimates through an inertial supervision learning scheme and a camera sensor fusion within a Recurrent Neural network aimed at optimizing unsupervised monocular trajectory estimates in systems with the presence of multiple cameras with non-overlapping fields-of-view (FOV).

Agradecimentos

Em primeiro lugar, queria agradecer ao meu orientador, Prof. Eduardo Silva pela excelente oportunidade que me proporcionou e pelas palavras de motivação ao longo desta etapa. Espero ter correspondido e continuar a corresponder às expectativas que depositou em mim. Agradeço também ao Prof. Aníbal Matos, pela disponibilidade que sempre mostrou para me receber e responder às minhas dúvidas.

Não posso deixar de mencionar os meus colegas investigadores do CRAS, nomeadamente a Sara, o Eduardo, o Tiago, o Ricardo, o Denis, o Caio, o Pedrosa, o Amaral e o Miranda. Obrigado pelo forma como me receberam e pelo excelente ambiente que se vive aqui dentro.

As minhas próximas palavras são para quem me trouxe até aqui, os meus pais Jorge e Fátima. As palavras não chegam para descrever a gratidão que sinto pela educação que me deram, pelo carinho, pela força e pelas condições que me proporcionaram.

À minha madrinha Celeste, por tudo o que fizeste por mim. Ao Hugo, por seres um exemplo para mim, me mostrares o caminho e incentivares a cada passo do caminho.

Aos meus avós, tios, primos e todo o resto da minha família, cada um à sua maneira importante nesta caminhada. Uma palavra especial para quem já não cá está, mas nunca partiu do meu coração.

À rapaziada do DMESM, foi um prazer partilhar esta caminhada convosco. 5 anos de pura partilha nesta montanha russa de emoções, não podia pedir mehor companhia nesta etapa.

Aos meus amigos Salvador, Kiko, Hugo, Filipe, Diana e Isabel, já sabem que são a família que eu escolhi, e se largos dias tem 100 anos, largas noites tem histórias que ficam para a vida. Obrigado especialmente por terem sido o garante da minha sanidade ao longo deste percurso.

E por último, quero agradecer à pessoa única e especial que é a minha namorada Ana. Entrámos já juntos nesta caminhada, e sem ti nada teria sido igual. Obrigado por seres a minha rocha e por sempre me motivares para ser a melhor versão de mim.

A todos o meu muito obrigado,

Bernardo Teixeira

"Our intelligence is what makes us human, and AI is an extension of that quality"

Yann LeCun

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUV	Autonomous Underwater Vehicle
BA	Bundle Adjustment
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRAS	Centre for Robotics and Autonomous Systems
DVL	Doppler Velocity Log
GPU	Graphical Processing Unit
IMU	Inertial Measurement Unit
INESC TEC	Instituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência
JPEG	Joint Photographic Experts Group
LSTM	Long Short-Term Memory
ML	Machine Learning
PNG	Portable Network Graphics
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
ROS	Robot Operating System
SBL	Short Baseline Acoustic Positioning Systems
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
USBL	Ultra-Short Baseline Acoustic Systems
VO	Visual Odometry
6-DoF	Six Degrees of Freedom

Chapter 1

Introduction

> "Aside from its importance to many branches of science, a knowledge of the oceans has a practical value for mankind. The intelligent development of our fishing industries, the laying of oceanic cables, the proper construction of harbor-works, oceanic commerce and navigation, as well as long-range weather forecasting, are all dependent on an understanding of the ocean."
>
> In The Last Cruise of the Carnegie (1932) by J. H. Paul.

1.1 Context

The ongoing evolution of underwater robotic vehicles, whether autonomous or remotely operated, is closely tied to its reliable application in potential risk carrying tasks or challenging environments for human operators. Such applications range from oceanographic monitoring or demining operations to bathymetric or topological mapping of ocean floors. Gathering and analyzing data from marine resource filled ocean floors has helped improve further research in marine biology, as well as fuel the recent growth in marine blue economy all over the world.

Autonomous Underwater Vehicle (AUV) technology development started in the 1970's and progressive improvements in computational efficiency, hardware size reduction and navigation accuracy have contributed to its increasingly importance for marine exploration, deep sea applications and mapping of ocean floors. To perform this tasks, robot navigation in complex unstructured environments is a crucial aspect of robotic solutions development.

There has been a substantial increase in the use of underwater robotic solutions, that are typically equipped with a plethora of heterogeneous sensors such as: acoustic sensors, laser rangefinders, Doppler Velocity Loggers (DVL), Inertial Measurement Units (IMU) or Global Positioning System (GPS). The combination of these sensors aims to provide precise and reliable robot localization and navigation. However, its application is limited when it becomes necessary for the robot

to remain submerse for long periods of time relying solely on perceiving sensorial information, since it acccumulates drift in navigation estimates.

Typical robotic application scenarios are prone to IMU/GPS failures, thus making it necessary to use alternative or complementary sensors. In the underwater context, since GPS is not available as a navigation solution due to electromagnetic waves being severely attenuated underwater, the use of Doppler Velocity Loggers (DVL) and Acoustic Localization methods (SBL/USBL) is commonplace. Doppler Velocity Logs work by obtaining the frequency distortion between emitted and received signals, denoted as Doppler Effect, while Acoustic Localization functioning principle is computing the propagation time of acoustic signals in water, and triangulating at least three emitters data to recover absolute localization. However, classical dead reckoning methods tend to drift substantially, thus making it necessary to fuse information from different sensors in order to achieve persistent autonomous navigation.

To tackle the issue of precision and reliability, scientists and researchers started exploring the possibility of including camera setups in underwater robotic solutions, so as to further expand upon robot autonomy capabilities.

One example of a task that can be performed through the use of visual camera setups is motion estimation based on visual information, denoted as Visual Odometry (VO). VO is an advantageous approach for computing navigation data, often outperforming classical dead reckoning methods(e.g. wheel encoder sensors in case of slippery terrain). With increasing levels of performance and accuracy, visual-based robotic navigation is gaining wider acceptance in the Robotics community, posing as a viable alternative to classical solutions.

Egomotion estimation, or self-motion estimation, commonly denoted as Visual Odometry (VO) in the Robotics community, is defined as the measurement of environmental displacement of the observer, usually referring to a camera system. The goal is to extract the 2D/3D movement of said moving camera from a sequence of images. This task is almost a prerequisite for many computer vision applications in mobile robots such as obstacle detection, autonomous driving or Simultaneous Localization and Mapping (SLAM). All these applications require a measure of the relative motion of the current camera frame with respect to the previous camera frame, in order to estimate the camera pose (position and attitude).

Though Visual Odometry algorithms have already undergone several years of research and development, with increasing levels of performance and accuracy, there are still many open challenges for visual-based robotic navigation applications. With this in mind, novel data-centric approaches have been surfacing, posing as a viable alternative approach to VO based geometric solutions.

1.2 Motivation

Given that robot navigation is a crucial aspect in the field of underwater robotics, an extensive research has been continuously devoted to solve the robot navigation problem and help achieve persistent autonomy for vision-based mobile robots, especially in unknown environments. Underwater mobile robotics application scenarios present uniquely challenging conditions for vision-based egomotion estimation due to several environmental factors, namely: the lack of appropriate lighting conditions, water turbidity, backscattering, lack of image texture and vignetting effect.

In addition to this frequent problems, the amount of sensorial information provided by visual system applications usually leads to a situation of information overload, by which not all the gathered data is adequately processed. The above referenced issues, along with the development and proliferation of cheaper Graphical Processing Units (GPU) alternatives, have prompted the research into data-driven methods such as deep learning for egomotion estimation. The field of deep learning applications for computer vision problems has had a tremendous surge in the last decade, posing as a viable and robust alternative to classical approaches.

State-of-the-art approaches , heavily reliant on geometric models, have shown to perform robustly on robotic applications. Yet, the advent of deep learning methods brings forward the expectation of improving upon visual task performance through the use of data-centric approaches. Experimenting, testing and evaluating the performance of such approaches in real operational data is therefore key to understand the degree to which they can provide accurate and robust performance in complex mission scenarios, as well as pose as alternative solutions for possible future implementations in robotic applications.

INESC TEC Centre for Robotics and Autonomous Systems (CRAS) has been conducting several projects in the domain of underwater robotics, often coming across the very same problems. The motivation for the proposal of this Msc Thesis is to understand the extent to which end-to-end data-driven methods can help tackle robotic visual navigation issues, in particular its robustness when being deployed in large-scale harsh operational environments, or under extreme lighting conditions.

The application and development of novel deep learning approaches will contribute to further advance the robotic solutions of ongoing projects like the UNEXMIN, TURTLE or VAMOS robots, currently being developed at CRAS [1] [2].

(a) EVA hybrid ROV

(b) Turtle Lander

(c) UNEXTMIN UX-1 AUV

Figure 1.1: Example CRAS robots

1.3 Objectives

This thesis main objective is to be able to robustly estimate robot 6-DoF pose and/or its egomotion in the underwater context using deep learning approaches. To do so, we can decompose this objective into the following secondary objectives:

- Acquisition of novel datasets for motion estimation using visual based methods of an AUV/ROV in underwater environment. This is to be achieved through extracting visual and inertial information from a CRAS robot, in particular the UX-1 robot. The goal is to build a comprehensive dataset that encompasses different texture environments and adresses all typical problems in the underwater context. Two different scenarios are to be taken into account: fully known CRAS pool environment and real operational UX1 mission scenario.
- Performance assessment of state-of-the-art data-driven end-to-end methods based on deep-learning approaches for Visual Odometry estimates. Performance shall be evaluated against its respective ground truth for two different tasks: Absolute Relocatization and Egomotion Estimation.
- Develop and test the fusion of visual information data with inertial data. Assess performance gains with Deep Learning architecture for sensor fusion, when compared with solely visual-based visual approaches.

- Evaluate all the implemented approaches via real robot data, taken in challenging underwater conditions.
- Contribute to the long-term enhancement of CRAS robots perception capability, particularly with respect to visual-based robot navigation solutions.

1.4 Contributions

During the course of this work, a bottom-top approach was followed, in order to guide and support the development of the thesis scientific work. This approach lead to the following contributions, namely:

- Perform a comprehensive review of state-of-the-art Deep Learning approaches for Visual Odometry applications, to assess the main advantages and/or disadvantages of the proposed methods in order to identify the main method bottlenecks and degenerate failure conditions. It is important to mention that most of deep learning VO state-of-the-art approaches, were mainly tested in Urban operational scenarios (car datasets), which poses a completely different challenge to these algorithms when comparing to the underwater VO estimation scenario.
- Use and test different underwater robot visual data, using the UX-1 robot [2], for the development and creation of novel image datasets, that allow the test and development of SOA and novel deep learning based Visual Odometry solutions, and access their feasibility for application in real robot underwater navigation scenarios i.e. CRAS pool and Urgeiriça mines.
- Evaluation of the performance of the most renown state-of-art deep learning VO estimation, using supervised and non-supervised VO estimation approaches in the UX-1 dataset sequences.
- Perform context fine tuning of the state-of-the-art methods, in order to improve results for VO estimation in underwater environment. Compare VO estimation results against ground-truth information, obtained by fusing Inertial and DVL sensors information.
- Development of a novel supervised deep learning approach by fusing visual and inertial sensor information within an artificial neural network. Comparison against previous obtained results.
- Preliminary development of an exploratory approach, consisted of a multi-camera pose estimate fusion Recurrent Neural Network for non-overlapping field-of-view monocular setups.

This work lead to the following scientific publication:

- Bernardo Teixeira, Hugo Silva, Anibal Matos and Eduardo Silva, "Deep Learning Approaches Assessment for Underwater Scene Understanding and Egomotion Estimation", OCEANS 2019 Seattle (Accepted for publication and inclusion in the Student Poster Competition)

1.5 Document Structure

This section serves the purpose of detailing the structure of this document.

In chapter 1 we introduced the research topic, describing the context and scope of application of this Msc thesis. We explained the motivation behind novel data-driven approaches to egomotion estimation problems and further described the objectives we hoped to accomplish in this thesis.

In chapter 2 we review related work on Computer Vision, mainly focusing on Visual Odometry applications. We review both monocular and stereo approaches, further separating feature-based and dense methods. Finally, we turn our attention to Deep Learning and the novel data-driven visual-based robot navigation approaches that have been flourishing amongst the Robotics community.

Chapter 3 introduces key fundamental concepts and governing principles aiming to contextualizing the reader to the research topic. We start by formulating the Visual Odometry problem, and proceed to cover different relevant topics ranging from the pinhole camera model to deep learning key concepts and techniques.

Chatpter 4 provides a review and assessment of existing algorithms that perform relocalization and egomotion estimation task using deep learning techniques. The novelty is the application of such methods to the underwater context, which to the best of our knowledge, has not yet been attempted.

In chapter 5, we present two novel deep learning architectures for improving upon global trajectory estimates obtained through concatenation of deep learning framework generated egomotion estimates. We propose to leverage the sequential nature of the data to enforce global pose optimization and discuss the results of both networks.

Chapter 6 discusses the overall performance of deep learning methods for visual-based robot navigation tasks, offering insight about potential applications on board of real robotic solutions. Further, it ponders upon future work to be done in the deep learning domain for computer vision applications.

Chapter 2

Related Work

2.1 Introduction

This chapter focuses on the thesis related work. It underlines the areas where Deep Learning research has provided significant results or promising potential. A brief overview of state-of-the-art research conducted over the past 30 years on the topic of Visual Odometry is provided, as well as an explanation of the technological motivations prompting the surge of data-driven methods such as deep learning for performing the task of egomotion estimation.

To do so, it starts by addressing the problem of recovering relative camera pose and obtaining motion information from a set of camera images, exploring geometrical approaches to the problem. Secondly, it details the functioning principles of methods that perform egomotion estimation using computer vision applications in the context of robotic solutions, such as Structure from Motion or SLAM systems. Finally, it focuses on novel data-centric approaches using Deep Learning, that aim to help tackle robotic visual navigation issues in situation where classical methods tend to fail or flat out struggle to perform. Common benchmark datasets used in the context of evaluating algorithm performance are also discussed in the last part of this section.

2.2 Retrieving Image Information

As a prerequisite for many robotic applications, it becomes necessary to accurately estimate self-motion of a robot's camera system. Given that input data are visual images, the motion estimation task is performed by identifying and measuring the relative displacement between consecutive images in a temporal sequence.

Recovering motion information has presented itself as a somewhat challenging task. Both the Computer Vision and the Robotics Communities have been conducting extensive research and presenting novel approaches and methods almost on a yearly basis.

In recent decades, many different motion estimation algorithms and approaches have been developed, matured and perfected. The taxonomy may be divided into two main categories based on camera setups: approaches using a monocular camera (e.g. Yamaguchi *et al* [3]) and approaches using a stereo camera (Nister *et al* [4]). These approaches can be further separated into methods that either use feature-based methods (e.g. Howard *et al* [5]) or dense (e.g., Engel *et al* [6]) methods. In this section, we review relevant prior work in all aforementioned categories.

2.2.1 Stereo Vision

The term stereo camera vision accounts for the fact that information is extracted from multiple cameras and the relative position of detected features can be directly measured by triangulation and used to derive motion. Given that information on the third dimension (i.e. depth) can be extracted from a single frame, the image scale can be immediately and instantaneously retrieved because the size of the stereo baseline (distance between the two camera lenses) is fixed and known, thereby resulting in an efficient and accurate triangulation process.

In order to facilitate correspondence in stereo camera setups, it is usual to employ a transformation process called stereo rectification [7]. The motivation behind this procedure is to determine a transformation of the left and right camera image planes so that epipolar lines become collinear and parallel to one of the image axes, hence simplifying the stereo correspondence problem through reducing it to a 1D horizontal search along the epipolar line in the new rectified image.

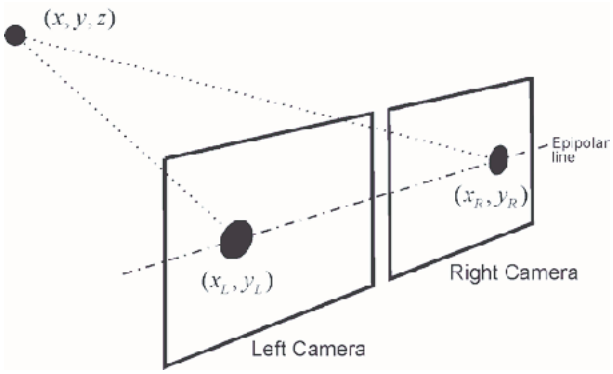


Figure 2.1: Stereo vision

The early work of Moravec [8] provided a description of the first motion estimation pipeline (whose main definitions remain basically intact today) as well of his renowned corner detector. This work formed a basis for further research in motion estimation using stereo vision systems, extended by means of correcting absolute orientation [9] and careful selection of key points through analyzing the curvature of auto-correlation function around feature peaks. Coupled with a robust least-squares motion estimation step around an outlier rejection scheme denoted as RANSAC [10], Cheng *et al* [11] developed the definitive VO implementation on board the Mars Rover (fig 2.4).

The downside of stereo vision applications is the need for an adequate and precise camera calibration, as it strongly impacts motion estimation accuracy. Also, stereo vision systems degrade to the monocular case when the stereo baseline is much smaller than the distances to the scene from the camera. Stereo vision becomes ineffective in this case, and monocular methods are recommended as per Scaramuzza and Fraundorfer findings [12] [13].

2.2.2 Monocular Cameras

The alternative to stereo vision is to use a single camera. In monocular VO systems, both the relative motion and 3D structure must be computed from two-dimensional data, hence the absolute scale is unknown. However, there are methods that allow for absolute scale recovery, though sometimes with limitations. Scale can be determined from direct measurements (e.g. the size of an element in the scene across images) or extracted from complementary sensors, such as IMU's or wheel odometry. For every new image, the relative scale and camera pose are computed between current and previous frames.

There are two publications worth mentioning: the seminal publication of Nister *et al* [4], with his 5-point algorithm, proposed the first real-time large scale VO system using monocular camera, using RANSAC for outlier rejection and computing upcoming camera poses through 3D-2D camera pose estimation. Corke [14] provided a different, biologically inspired approach, using an omnidirectional camera and capturing optical flow (the pattern of apparent motion of objects in a scene caused by the relative motion between an observer and a scene).

Independently of the camera setup, the ultimate goal is to extract and process information from visual image inputs. In the next sections, we will further discuss image information retrieval, expanding on the difference between feature-based and dense methods, as well as detailing the guiding principles of novel application for each group.

2.2.3 Feature-based Methods

Feature-based methods for visual motion estimation are based on salient and repeatable features that are tracked across multiple frames. The process starts with the detection and matching of such features and the camera pose is computed based on a set of feature observations.

A feature is defined as an image pattern that is significantly different from its immediate neighboring pixels, either in term of intensity, color or texture. In Visual Odometry applications, the most common traits to search for are usually edges, corners or blobs. An edge is defined as a place of rapid change in the image intensity function and the intersection of two or more edges is denoted as a corner. Blob is the computer vision term that represents a group of connected pixels that have similar intensity values within the blob but have different values from the ones surrounding it.

Features possess interesting characteristics that help explain why it is so appealing to design geometric models around them, namely:

- Quantity: There can be hundreds or thousands of features in a single image.
- Repeatability: features can be redetected in the next images, allowing for feature matching.
- Locality : Features are local, and theoretically more robust to occlusions or visual clutter.
- Distinctiveness : Through combination or correlation of features, it is possible to differentiate between a large database of objects.
- Efficiency: Feature-based methods are proven to achieve real time performance.
- Robustness: Features are robust to noise, blur or the presence of artifacts in the image.

In addition to these properties, a good feature detector should focus on detecting features that are invariant to both photometric changes (i.e. changes in illumination, brightness, exposure, etc) and geometric changes like rotations, scale or distortions.

In egomotion estimation, consecutive images will show changes due to differences in rotation, scale and illumination and possibly image noises. Being based on salient key points, these methods are theoretically more robust to these effects. Novel scale invariant methods such as Scale Invariant Feature Transform (SIFT) [15], Speeded Up Robust Feature (SURF)[16] address this very same issues and create invariant descriptors, even in cases of abrupt rotational or scale changes or even under non-homogeneous lighting conditions. Real time computation of high resolution imagery prompted the use of faster computational descriptors like ORB [17] or BRIEF [18].

The feature transformation step greatly reduces the computational time needed to process information and allow for smoother real-time implementations. For this reason, feature-based methods dominate the realm of solutions to the pose estimation problem.

However, feature-based methods process only the information contained in the detected features, effectively discarding a great amount of visual clues about the scene that could otherwise be helpful in some applications. Furthermore, typical approaches require huge amounts of computation concerning scale and rotational invariant descriptors and are thus prone to show weaknesses in operational environments.

2.2.4 Direct/Dense Methods

Algorithms for aligning images and estimating motion in video sequences are widely used in computer vision since the early days of digital cameras and primitive image stabilization features. Dense methods are methods by which the intensity information of all image pixels or subregions of it is computed in order to perceive motion in sequential images.

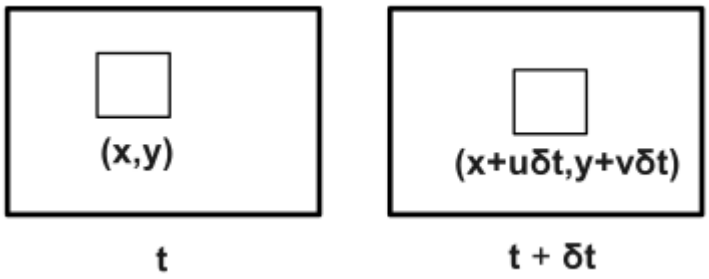


Figure 2.2: Brightness shift of an 2D image pixel representation. The image pattern at position (x,y,t) is the same of position $(x+u\sigma t, y+u\sigma t, t+\sigma t)$. Image courtesy of [19]

The most general approach, denoted as optical flow, involves minimizing the brightness or color difference between corresponding pixels over a time frame.

Optical flow methods are often divided in three main categories [20] [21]:

- **Differential methods:** The motion is computed from spatio-temporal derivatives or filtered versions of the image [22] [23]. [24]
- **Frequency methods:** These methods work by applying spatio-temporal filters in the frequency domain [25] [26].
- **Correlation methods:** Correlation based methods attempt to find matching image regions by maximizing some similarity measure between them, all under the assumption that the image has not been overly distorted over a local region for a short period of time [27] [28] [29].

Dense methods are more suitable to tackle small scale motion displacement estimation problems but often pale in comparison to feature-based counterpart methods when analyzing large scale robotic applications self-motion.

2.3 Egomotion estimation

The problem of estimating a vehicle self-motion based on visual input alone first saw light in the early 1980's. The goal was to develop a robust visual navigation system for the NASA Mars Rover, capable of providing the planetary rover the capability to measure 6-Degrees-of-Freedom (6-DoF) motion , essentially reducing the risks of classic odometry failure due to wheel slippage and uneven terrain.

In the landmark paper by Nister *et al* [4], the term Visual Odometry (VO) was first introduced and a Visual Odometry pipeline was presented (Fig. 2.3) .

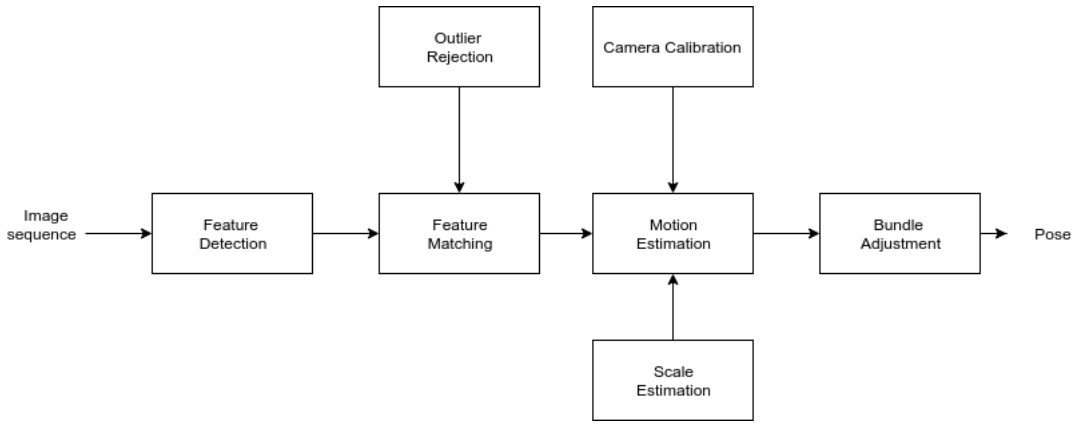


Figure 2.3: Typical Visual Odometry pipeline: Usual blocks present in Visual Odometry applications

Visual Odometry is defined by the process of estimating self-motion of a robotic application using only image measurements from a single or multiple cameras. The principle of operation of VO is estimating robot pose through examining changes in sequential images sets, induced by motion. It has demonstrated to be more advantageous with respect to wheel odometry under some circumstances and is a useful supplement to other common navigation systems such as Inertial Measument Units (IMU) and Global Positioning Systems (GPS). Typical robotic application scenarios are prone to IMU/GPS failures, thus VO has the utmost importance and extensive research has been conducted in the topic [30] [19].

Recent work on stereo VO has shifted away from planetary rover applications, broadening the spectrum of Visual Odometry applications to the development of novel autonomous vehicles and also more commercial applications in the automotive industry. This apparent proliferation of

(a) Mars Rover (b) Car applications

Figure 2.4: VO applications: early research and first applications were closely tied to space exploration. In recent years, an ever-growing number of commercial applications have surfaced all over the world, especially in the automotive industry

vision-based systems in commercial applications is, in part, due to recent advances by Scaramuzza *et al* [31] that allow a much faster motion estimation by applying physical constraints to help reduce model complexity.

In the last couple years it has been suggested that vehicle motion could benefit from relying less on geometry-based approaches and carefully calibrated camera motions, instead adopting a learning scheme for capturing vehicle dynamics and camera motion. Machine learning techniques [32] have been gaining favor, in particular thanks to the advent of more precise and robust approaches such as Deep Learning, which have proven capable of being a reliable alternative to conventional methods. In the next section, we present novel approaches to the motion estimation problem which use Deep Learning architectures to perform such tasks.

2.4 Deep Learning for Computer Vision

Both feature-based and dense methods for Visual Odometry estimation have made great success in the past decade, demonstrating ever-growing performance on both relocalization and motion estimation. However, they still face many challenging issues, in particular when being deployed in large scale robotic applications and facing complex environment application scenarios.

Usually, the previously discussed model based methods represent manually designed features and search for the best pose which matches the detected features between consecutive image frames. On the contrary, Deep Learning methods can effectively learn good representations of input images from massive amounts of data, not requiring manual extraction of features nor extensive or complex geometric formulations for modeling the environment. In recent years, Deep learning methods have shown good capability for cognitive and perceptual tasks in Computer Vision applications whether by analyzing unknown features, image depth or even egomotion between image frames. For more information on Deep Learning and a brief overview of key concepts we refer both to chapter 3 of this thesis and Ian Goodfellow *et al* book on Deep Learning [32].

In this section, we will review state-of the art Visual Odometry implementations running on top of Deep Learning architectures, analyzing key contributions for the research topic and nuances between emerging approaches to the problem.

2.4.1 Depth Estimation

Depth estimation methods take advantage of camera displacement or difference in the apparent position of an object viewed along two different lines of sight to estimate depth.

Early work by Eigen *et al* [33] proposed a supervised method for depth estimation with a ground-truth depth map and a scale-invariant error as a cost function for training. The work was further extended by further integrating convolutional neural networks improving accuracy and efficiency on both segmentation tasks and depth estimation. CNN-SLAM [34] is a proposed monocular SLAM system that relies on convolutional neural networks solely to estimate depth, recovering pose and graph optimization from conventional feature-based SLAM. This approach demonstrated that Deep Learning architectures can also work hand-in-hand with vision-based systems, improving upon overall robustness and accuracy of said algorithms.

Unsupervised schemes have recently emerged, also posing as viable alternatives. Garg's idea [35] was to use CNN's to predict the depth map for the left input image, reconstructing the left image from the right image and using the photometric reconstruction error (2.1) between the original left image I and the new synthetized left image I' in the training phase of the algorithm.

$$E = \sum \left\| I - I' \right\|^2 \tag{2.1}$$

SfMLearner [36] is a solution that established an influential framework for Deep Learning for Visual Odometry research. It uses a monocular image sequence in order to estimate depth and pose simultaneously in an end-to-end unsupervised manner, through enforcing geometric constraints between image pairs in the view synthesis process. Based on this framework, there were several novel architectures developed. SfMlearner++ [37] improved upon the results in both depth and pose estimation by using the Essential matrix, obtained using Nister's Five Point Algorithm [38], to enforce epipolar constraints on the loss function, effectively discounting ambiguous pixels.

GeoNet [39] is a similar approach, a jointly unsupervised learning framework for monocular depth, optical flow and egomotion estimation that decouples rigid scene reconstruction and dynamic object motion, making use of this knowledge to further tailor the geometric constraints to the model Vijayanarasimhan *et al.*[40] presented SfM-Net, innovating through adding motion masks to photometric losses to jointly estimate optimal flow, depth maps and egomotion.

2.4.2 Global Pose Estimation

Localization is a crucial component for autonomous systems development, that enables a robot to determine where it is on an environment, which serves as a precursor to any type of action execution or planning.

The main purpose of data-driven pose estimation is to estimate pose without explicitly modeling the camera motion. PoseNet [41] was the first instance of CNN usage for pose estimation, starting from a supervised scheme with a 6-DoF pose ground-truth. Making use of geometry to design meaningful constraints to the loss function [42] proved to yield significant improvements to method performance and accuracy. This method showed very robust performances in relocalization tasks and was further extended to support both color and depth inputs, improving upon its accuracy in challenging environments, such as night-time.

RCNN architectures have been gaining favor in the past years, exploring either temporal dynamics or spatial analysis of image sequences, thereby reducing the uncertainty of pose estimation and generally improving upon method performance. The introduction of LSTM units to neural network design as showcased in [43] proved to improve results in localization tasks making use of structured correlation in feature space using LSTM units.

The application of deep RCNN's to Visual Odometry bypass the need for almost all blocks in the conventional VO pipeline (see Fig.2.3), allowing for end-to-end pose inference as can be seen in Fig.2.5.

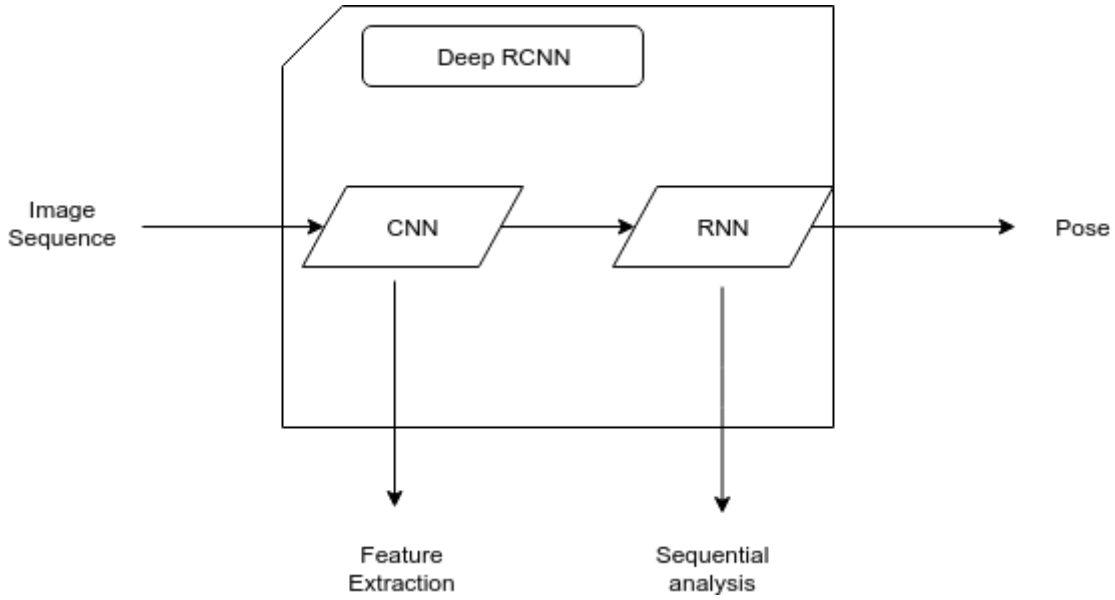


Figure 2.5: Proposed Deep Learning for Visual Odometry end-to-end methods pipeline. Deep RCNN takes a sequence of RGB images as input, automatically learning features by CNN, capturing temporal dynamics in its RNN component and outputting pose estimations

2.4.3 Egomotion Estimation

Building upon the success of absolute pose estimation, the egomotion between consecutive image frames can also be estimated with the use of deep neural architectures inspired by geometric models. The key principle is that for the egomotion estimation task we are interested in capturing the motion underwent by the camera system between consecutive images rather than just determining the position and attitude of the observer. FlowNet [44] and its successive iterations garnered immense attention as a reliable deep learning framework for learning optical flow and paved the way for early egomotion estimators. Wang *et al.* proposed a monocular visual odometry system called DeepVO [45], which trains a RCNN to estimate camera motion in an end-to-end fashion, inferring pose directly from a sequence of raw RGB images in a video clip while bypassing all usual modules in the conventional VO pipeline. The advantage of such approach is to simultaneously factor in both feature extraction and sequential modelling through combining CNN's and RNN's.

As labeling data in large scale significantly hinders the application of supervised learning methods to robotic applications, Li *et al* proposed UnDeepVO [46], a monocular system that uses stereo image pairs in the training phase for scale recovery. After training with unlabeled stereo images, UnDeepVO can simultaneously perform visual odometry and depth estimation with monocular images.

Valada *et al* [47] proposed a novel architecture that encompasses both global pose localization and a relative pose estimation, jointly regressing global pose and odometry and learning inter-task correlations and shared features through parameter sharing. This method is denoted as Deep

Auxiliary Learning.

Visual Odometry methods are particularly sensitive to rotation errors, as small early drifts can have a large influence on final trajectory pose estimates. Peretroukhin [48] proposed HydraNet, a deep learning structure aimed at improving attitude estimates, able to be fused with classical visual methods. Through regressing unit quaternions, modeling rotation uncertainty and producing 3D covariances, HydraNet manages to improve visual algorithms at predicting 6-DoF pose estimates (position and attitude).

Another application where Deep learning architectures are currently being tested on is sensor fusion. VINet [49] is a proposed framework that fuses pose estimates from DeepVO [45] with inertial data, showing comparable performance to traditional fusion systems. The same method was also adopted to fuse other kinds of information such as magnetic sensors, GPS, INS or wheel odometry. [50] [51]. Sensor fusion can be easily incorporated in deep learning architectures and jointly trained end-to-end with pose regression, thus making a potentially interesting solution for Visual Odometry applications as it can be used for a wide variety of purposes (e.g. recovering absolute scale on monocular camera systems).

2.5 Benchmark Datasets

Deep learning methods usually require massive amounts of data in order to properly train its neural architectures. This is particularly true in robotic applications, since autonomous systems typically operate in completely unknown environments or under extreme lighting condition. As so, the availability of large scale datasets is crucial for further development of deep learning methods. In this part, we review existing datasets usually adopted for evaluating deep learning related visual tasks.

The KITTI dataset [52] is usually regarded as a complete and comprehensive dataset for visual applications. It consists of outdoor environment images of a driving car, with multiple sensor data, best suited for egomotion estimation tasks. Similarly, the EuRoc MAV [53] also presents the same characteristics and also outdoor imagery but gathered by a flying robot.

When it comes to indoor imagery, the TUM dataset [54] is highly regarded as a comprehensive benchmark for depth estimation. It was collected through the use of an hand-held RGB-D camera in an indoor environment. Together, these datasets form a comprehensive basis for method evaluation, whether for depth or egomotion estimation.

For relocalization tasks, 7-scenes [55] dataset or Cambridge Landmarks [41] are best suited for evaluating method performance.

However, the focus of the thesis is in the underwater context. In the scope of underwater robotics there are no universally recognized datasets that fulfill either depth or egomotion estimation task evaluation requirements. Furthermore, the objective of this thesis is to evaluate the viability of

data-driven methods for motion estimation in the context of CRAS ongoing robotic applications. With this in mind, we developed a novel dataset comprised of underwater imagery acquired with the CRAS robot UNEXMIN UX1 robot, in both a fully known controlled environment and operational mission scenarios. For more information on our novel dataset, see chapter 4.

2.6 Summary

In this chapter, a study of visual-based navigation methods was performed, denoting the motivation behind the path from classical feature-based or direct/dense visual methods to learning-based data-centric approaches.

The scenarios where classical visual-based methods tend to fail, or at least do not provide the consistency and robustness required for persistent and accurate robot navigation prompted the development of data-centric approaches as an attempt to overcome such difficulties. For our use case, since we are focusing on the broader underwater context and particularly in this work on underwater flooded mines, deep learning method evaluation and assessment becomes especially relevant, as it could become a really interesting application for such scenarios

It is important to note that, at the moment, classical visual methods still dominate the realm of applications for robotic navigation solutions, especially due to its higher technological maturity, accuracy and robustness. However, novel deep learning powered architectures have show comparable performance in some situations, and it is reasonable to assume that deep learning methods will be able to compete with visual methods in the future, benefiting from more technological establishment and hardware enhancement allowing for better real-time performance on board the robot.

In order to bring some closure to this Related Work chapter, the main thought worth withholding is that deep learning based methods are becoming more and more reliable as years go by, posing as a viable alternative to classical Computer Vision methods for a wide range of different tasks. So, it becomes then necessary to evaluate whether specific robotic solutions can benefit from such approaches, possibly enhancing the perception capabilities of some robotic applications.

Chapter 3

Fundamentals

3.1 Introduction

This chapter begins by formulating the Visual Odometry problem and describing some of the key concepts and governing principles required to the thesis research topic. It aims to cover fundamental aspects of both Computer Vision and Deep Learning, focusing on some essential building blocks upon which different Egomotion Estimation techniques and methods are based. For each topic covered in this chapter, further references are provided, and the understanding of the basic concepts presented in this chapter will be important to the explanation of this thesis implementation and results.

3.2 Formulation of the VO problem

An agent is moving through an environment and taking images at discrete time intervals k. For monocular camera setups, the camera coordinate frame can be assumed as the agent's coordinate frame and is usually denoted as $I_{0:n} = I_0, ..., I_n$. In stereo vision systems, there is a left and a right image at every time instant, denoted as $I_{l,0:n} = I_l, 0, ..., I_l, n$ and $I_{r,0:n} = I_r, 0, ..., I_r, n$. The left camera is usually used as the origin, without any loss of generality.

Two camera positions at consecutive time instants are related to one another by rigid transformations $T_{k,k-1}$ defined as followed:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

$R_{k,k-1}$ represents the rotation from frame *k-1* to frame k and $t_{k,k-1}$ is the translation vector.

Additionally, a set of camera poses $C_{0:n}$ can be computed to contain information about camera transformations with respect to the initial frame k=0. Current pose C_n can be computed from the

previous pose and the rigid transformation between camera frames:

$$C_n = C_{n-1}T_n \tag{3.2}$$

The main task in Visual Odometry applications is to calculate image transformations $T_{k,k-1}$ and then concatenating the transformations to recover the full trajectory $C_{0:n}$ of the camera. This means VO systems recover the path incrementally, pose after pose.

As stated in chapter 2, there are two main approaches to compute the relative motion $T_{k,k-1}$: dense methods, which use the intensity information of all the pixels in the two input images, and feature-based methods, which only use salient and repeatable features extracted (or tracked) across the images.

In the VO pipeline, previously mentioned in Fig. 2.3, for every new image or stereo image pair I_k, a set of features f_k is extracted. The next step is to find correspondences between feature sets on time instances k and $k-1$. This can be accomplished either by finding features in one image and tracking them across the following images through prediction and correlation local search techniques or by independently detecting features in each image and find correspondence on the basis of similarity metrics (use of descriptors like SIFT, SURF or ORB) [15] [16] [17].

Motion estimates can be computed from transformations T_k between consecutive frames and corresponding feature sets f_{k-1} and f_k. Motion estimation correspondence models are further detailed in section 3.5.

The final step in VO systems is usually a local optimization block, consisting of iterative refinements over the last n poses. This step, commonly denoted as bundle adjustment [56] [57], aims to provide a more accurate motion estimation of local trajectories by minimizing re-projection errors between camera poses.

3.3 Perspective Camera Model

Camera models map 3D world point spaces into 2D image representations. The most common camera models for image interpretation and analysis are perspective camera models and are focusing our attention on the pinhole projective model.

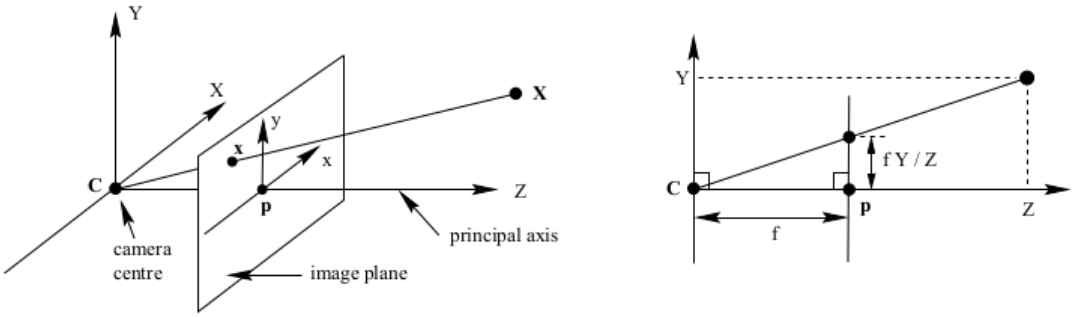


Figure 3.1: Pinhole Camera Model

In this model, a point in Cartesian space with coordinates $X = (X,Y,Z)^T$ is mapped on the point on the image plane where a line joining the point X to the centre of projection meets the image plane (Fig. 3.1).

Usually in computer vision notation, parameter f corresponds to the focal distance and parameters (cx,cy) represent the principal point (p) coordinates, which is the place where the principal axis intersects the image plane in pinhole camera model. The camera parameters like the focal distance f , principal point p, and skew s which is the angle between the x and y sensor axes, need to be known for the model to be applied. Together, these points form matrix K, which is the camera calibration matrix and defines the camera intrinsic parameters :

$$K = \begin{bmatrix} f & s & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

In Visual Odometry applications, there is sometimes a necessity to apply transformations between camera frame and world frame. Euclidean transformations are represented by a rotation matrix R in the 3D space and a translation vector t. Thus, camera projection matrix becomes:

$$P = K[\, R \,|t\,] \tag{3.4}$$

For further information on this topic, the reader is invited to [58][59][60].

3.4 Epipolar Geometry

Epipolar geometry is essentially the relation between images of a single point (**P**) captured from two different viewpoints (see Fig. 3.2). This concept can both be applied to the stereo case (two cameras simultaneously viewing the same scene) as well as to a single camera taking pictures standing from different viewpoints (monocular camera setups). There are a few concepts worth explaining:

- **Epipole:** The point of intersection of the line joining the camera centers (stereo baseline for the stereo case) with the image plane.

- **Epipolar plane:** The epipolar plane is formed by the world point **P** and the two camera centers, therefore containing the baseline.

- **Epipolar lines:** The intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole.

Exploring these epipolar geometry concepts, we come across a very useful definition: the epipolar constraint is a key geometric concept that limits the correspondence of a given point in image I_l to lie along a line in the other image I_r [60]. This correspondence is key to establishing point correspondence between images since it reduces potential correspondence points from the image plane to just the points in an epipolar line.

The Fundamental matrix (F) is a function of the camera intrinsic parameters (K), and also encodes the relative camera poses between the different viewpoints and can be given by:

$$F = K^{-1}R[t]_x K \tag{3.5}$$

Another way of encompassing information of the epipolar constraint is the Essential matrix [61]. The Essential matrix (E) is a specific application of the fundamental matrix to the case of calibrated cameras (i.e. intrinsic parameters known). The Essential matrix has 5 degrees of freedom and is defined by three rotational and two translation parameters

In summary, the Fundamental matrix (F) and a point in one image define an epipolar line in the other image on which its correspondent point must lie. The Essential matrix (E) encodes the relative pose between the two camera frames, and their pose can be extracted with some uncertainty only in translation displacement, which is scaled by an unknown factor. The geometric relations provide valuable information about scenes to the visual egomotion estimation problem, and are a basis for numerous Visual Odometry applications, both for stereo and monocular setups.

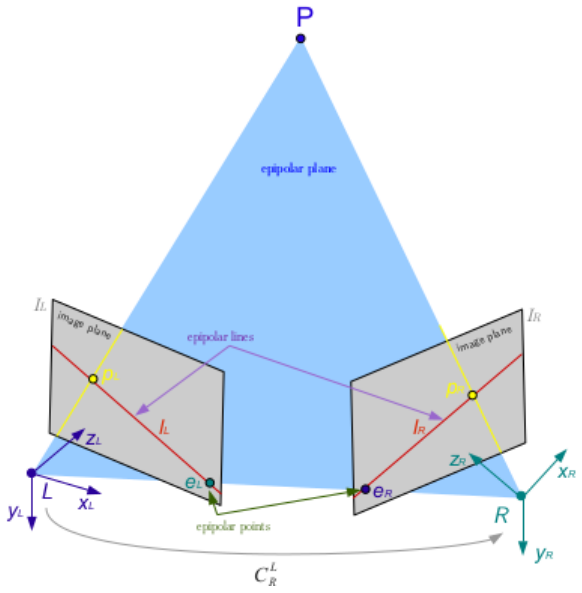


Figure 3.2: The world point P and the two cameras centers form the epipolar plane. The intersection of the epipolar plane with the image plane forms the epipolar lines. Image courtesy of [19]

3.5 Egomotion Estimation

Motion estimation is one of the core blocks in Visual Odometry systems. In order to perceive camera self-motion, VO systems make use of aforementioned spatial transformations between images I_k and I_{k-1}.

Using features sets f_k and f_{k-1} from consecutive images I_k and I_{k-1}, there are three methods for computing the transformation $T_{k,k-1}$:

- **2D-to-2D:** Feature sets f_k and f_{k-1} are specified in 2D image coordinates.
- **3D-to-3D:** Feature sets f_k and f_{k-1} are specified in 3D. This technique is available in stereo vision, and consists on determining the aligning of the two 3D feature sets in stereo image pairs.
- **3D-to-2D:** In this case, f_{k-1} is a 3D feature set and f_k is the correspondent 2D re-projection onto the I_k image. In monocular VO systems, 3D structure need to be reconstructed from at least two adjacent camera views I_{k-2} and I_{k-1} and matched to I_k image features.

3.5.1 2D-to-2D: Motion from Image Feature Correspondence

The Essential matrix E, which encompasses rotational and translational information , can robustly be computed from 2D-to-2D feature correspondences through the epipolar constraint, as per Nister's proposed five-point algorithm [38].

From the Essential matrix, we can extract the rotational and translational parts, through computing the Singular Value Decomposition (SVD):

$$E = USV^T \tag{3.6}$$

This equation returns four possible solutions for the R, t pair, but the correct representation can be easily identified by means of triangulation of a single point (i.e. only in one of the four possible solutions the point is front of both cameras).

As the absolute scale of the translational component cannot be computed in the essential matrix, relative scales must be computed from measured distances between point pairs in consecutive images in order to recover the trajectory of an image sequence.

The algorithm for egomotion estimation from 2D-to-2D correspondence works as follows:

Algorithm 1 Motion estimation from from 2D-to-2D feature correspondence

- Acquire new image I_k
- Extract and match features between I_k and I_{k-1}
- Compute essential matrix (E)
- Extract R,t from the essential matrix and form Euclidean transformation $T_{k,k-1}$
- Compute relative scale
- Concatenate camera pose transformation by performing $C_k = C_{k-1}T_{k,k-1}$

3.5.2 3D-to-3D: Motion from Structure Correspondence

In the case of 3D-to-3D motion estimation, the task is to find the transformation $T_{k,k-1}$ that minimizes the L2 norm distance between features (i) with their respective 3D coordinates between consecutive frames ($\hat{X}_k^i$ and $\hat{X_{k-1}^i}$)

$$arg^{min}_{T_{k,k-1}} \sum \left\| \hat{X}_k^i - T_k \hat{X}_{k-1}^i \right\| \tag{3.7}$$

As in the previous case, there is more than one solution to this problem and the correct solution must be recovered through triangulation. In this case, we decouple translation from rotation, computing translation through SVD and rotation as the difference from centroids of 3D features.

The algorithm for egomotion estimation from 3D-to-3D structure correspondence works as follows:

Algorithm 2 Motion estimation from 3D-to-3D structure correspondence

- Acquire new image stereo pair $I_{l,k}$ and $I_{r,k}$
- Extract and match features between $I_{l,k}$ and $I_{l,k-1}$
- Triangulate the previous features for every stereo pair
- Concatenate camera pose transformation by performing $C_k = C_{k-1}T_{k,k-1}$

3.5.3 3D-to-2D: Motion from 3D Structure and Feature Correspondence

Nister [4] has shown 3D-to-2D to be more accurate than 3D-to-3D. This is because instead of minimizing the distance between features on consecutive images, we focus on minimizing the reprojection error.

Motion can be estimated both from stereo images and from the monocular case, by triangulating three view points. In a stereo system, the 2D point can be obtained by aligning the left and right images whereas in the monocular case, the algorithm processes the triangulation between feature matches in I_{k-2} and I_{k-1} and subsequently matches to the new captured frame I_k

P3P [62] is a commonly used robust method for 3D-to-2D motion estimation, even in the presence of outliers, that computes camera pose and orientation in the world reference frame from three 3D-2D point .

The algorithm for 3D-to-2D egomotion estimation can be synthesized as followed:

Algorithm 3 Motion estimation from from 3D-to-2D feature correspondence

Initialization:

- Capture two frames I_{k-2} and I_{k-1}
- Extract and match features
- Triangulate features between them

For every iteration:

- Capture new frame I_k
- Extract features and match features with previous frame I_{k-1}
- Compute camera pose and orientation from 3D-to-2D matches
- Triangulate feature matches between I_k and I_{k-1}

3.6 Pose Parametrizations

The goal of every Visual Odometry system is to adequately process visual information in order to compute the camera transformation in relation to the previous image frame. As formulated in section 3.2, we are interested in obtaining a set of camera poses containing the information of relative transformations in camera position and attitude. These can then be concatenated to represent the full trajectory in an image sequence.

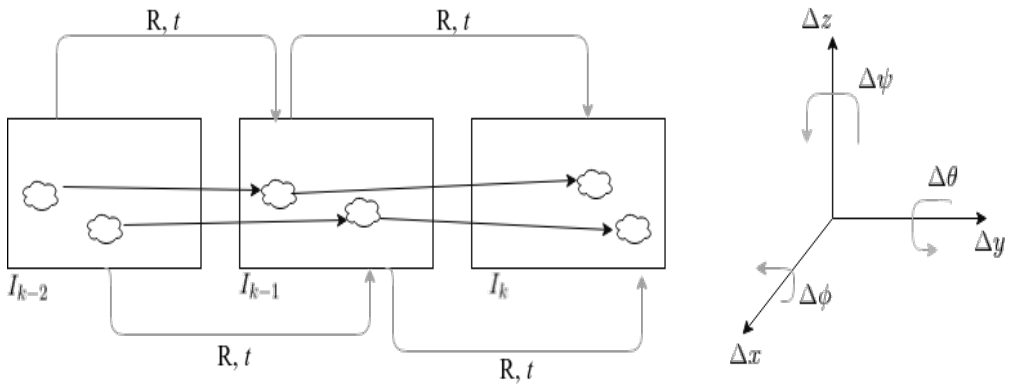


Figure 3.3: R,t parameter extraction

The literature presents different pose estimate parameterizations for 6-DoF pose (position and attitude). While the position of an object is naturally represented by a 3-dimensional vector without much debate, such consensus does not exist regarding its orientation representation. In this section we review key attitude representations that serve as a starting point for the work of this thesis.

3.6.1 Rotation Matrix

The rotation matrix, also called the special orthogonal group SO(3), is the most generic form for representing a linear transformation on an Euclidean space used to rotate point coordinates from one reference frame to another.

Considering two frames A and B and p as position of an object relative to frame A. Then the position of the object with respect to frame B is given by:

$$p_B = R_B^A p_A \tag{3.8}$$

where R_B^A represents the rotation undertaken by point p from one frame to another. This means nine scalars are used to represent rotations in 3D space (i.e. 3x3 matrix).

Coupled with a translation vector in 3D space ($\in R^3$), we can represent the Special Euclidean group SE(3), an all encompassing space for rigid transformations.

3.6.2 Euler Angles

However, it is possible to parametrize the very same rotation using much less parameters. Euler Angles Representation is defined by the angles that each axis of frame B needs to rotate, in a sequential order, to coincide with a reference frame A.

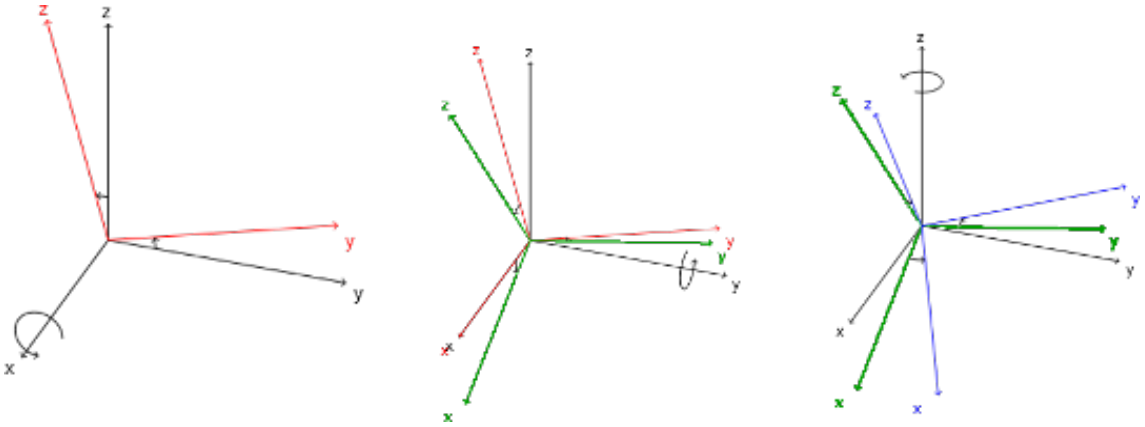


Figure 3.4: Euler Angles representation

This representation can be more intuitive and easy to interpret physically. However, it suffer from singularities in mathematical formulations (e.g. such as gimbal-lock in Roll-Pitch-Yaw). Notation can also be problematic, as there exist twelve possible sequences of rotation axes, giving rise to potential confusion due to notation ambiguity that can be detrimental for cross-implementation integration.

3.6.3 Quaternions

Quaternions are R^4 unit vectors that can also be used to represent rotations very efficiently. It can also provide a more restricted parameterization of the output space when compared to the entirety of the transformation matrix set. A quaternion can be defined as:

$$q = (w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}) \tag{3.9}$$

Quaternions [63] are sometimes an advantageous representation in light of its computational efficiency (simple number operations are computationally less costly than sines and cosines) and because unlike Euler Angles it does not have any discontinuity . Interpolating between rotations and especially chaining multiple transformations is therefore a way smoother process using quaternions.

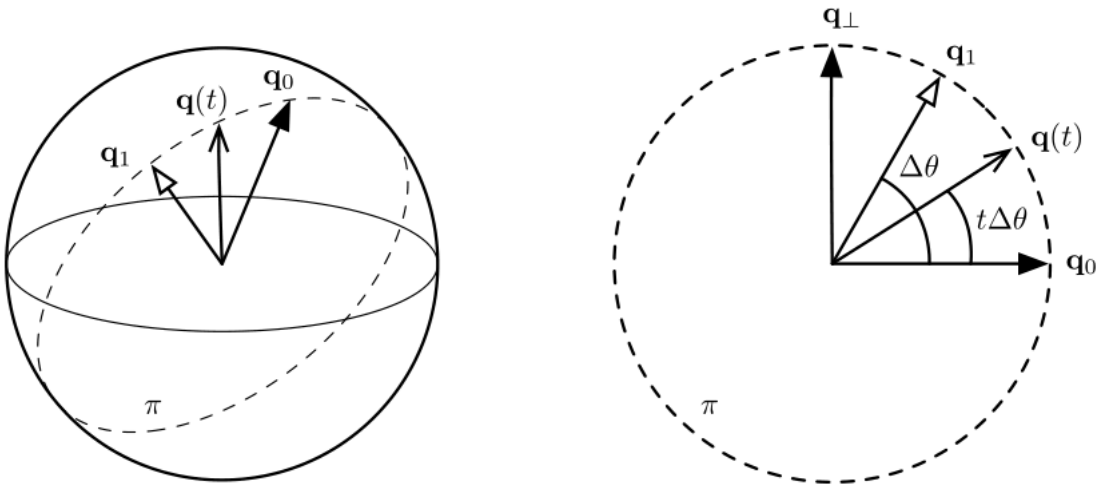


Figure 3.5: Quaternion Unit Sphere

3.7 Deep Learning

3.7.1 Motivation

Machine learning is defined as the scientific study of algorithms and applied statistical models that allow computer systems to estimate otherwise complicated functions, progressively improving upon its performance on a specific task.

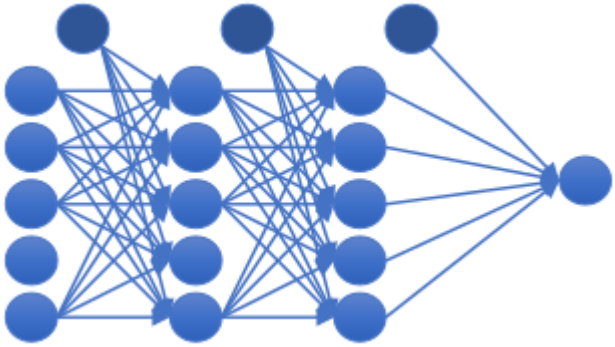


Figure 3.6: Neural network

Deep learning is a specific kind of machine learning. Its onset stems from the failure of traditional machine learning algorithms to generalize well on some artificial intelligence tasks and deep learning methods are, by design, more fit to overcome these and other obstacles.

One of such tasks is Computer Vision. Computer vision has traditionally been one of the most active research areas for deep learning, due to it being intrinsically challenging to implement simultaneously robust and efficient computer algorithms. The ongoing rise in large scale dataset

availability, as well as the proliferation of more capable GPU's is fueling the evolution of deep learning-based methods for computer vision, improving upon their performance and accuracy.

In this sub-section we explore some key concepts of deep learning as to contextualize the reader to some deep learning techniques that will be explored as the focus of the implementation of this thesis. For further detail on deep learning techniques and overall machine learning concepts, the reader is invited to Ian Goodfellow [32].

3.7.2 Supervised vs Unsupervised schemes

Within the field of machine learning, there are two main classifications for algorithm architecture: supervised and unsupervised.

Supervised learning schemes are implemented by using a ground truth, i.e. we have prior knowledge of what the output values for our training samples should be. Therefore, the goal of supervised learning is to learn a function that, given a sample of data and desired outputs, best approximates the relationship between input and output observable in the data.

Unsupervised learning schemes are built with the intent of learning the inherent structure of our data without the use of explicit labels. Here the task consists in grouping unsorted information according to similarities, patterns and differences without any prior training. In situations where it is either impossible or impractical for a human to propose trends in the data, unsupervised learning can provide initial insights that can then be used to test individual hypotheses.

3.7.3 Concepts and Techniques

There are a few key principles and techniques worth mentioning about Deep Learning in the scope of this thesis:

- **Tensor:** A tensor is an array of numbers arranged on a regular grid with a variable number of axes.
- **Activation Function:** This block defines the output of a neuron, deciding whether it should fire or not. It maps the resulting values of neuron computations to a preset desired range (depending upon the choice of activation function). Functions like sigmoid and ReLU are commonly used in neural networks to help build working models, but proper choice of activation functions is application specific.
- **Feature Map** A feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. A produced vector passes through an activation function, forming a tensor that is then used as an input for the next layer.

- **Loss function:** A loss function is a measure of how good a prediction model does in terms of being able to predict the expected outcome. Optimization of weights and other parameters are closely tied to the choice of an adequate loss function. There is not a single loss function that works for all kind of data. It depends on a number of factors including the presence of outliers, choice of algorithm, time efficiency of gradient descent, ease of finding the derivatives and confidence of predictions. Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^e)^2 \tag{3.10}$$

- **Dropout:** Dropout is a machine learning technique designed to help prevent model overfitting. Simply put, dropout refers to not considering a random batch of neuron units during the training phase .

3.7.4 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are one of the most popular deep learning architectures. Its design is biologically inspired, drawing inspiration from neurocientific research on mammalian vision systems, as per Medicine Nobel laureates Hubel and Wiesel findings [64]. Therefore, it is safe to assume that CNNs could present as an adequate technique to process image data, as claimed by recent research on the topic detailed in the previous chapter.

However, it is worth noting that neuroscience has told us relatively little about the process of training and tuning hyperparameters in a Convolutional Neural Network. The detachment from biologically inspired designs and constraints, and progressive integration of more mathematical and statistical focused approaches further improved upon the early concepts and matured into a specialized neural network architecture that works pretty well with data that has a clear grid-structured topology (e.g. image data, as a 2D pixel grid) and is able scale such models to very large quantities of information.

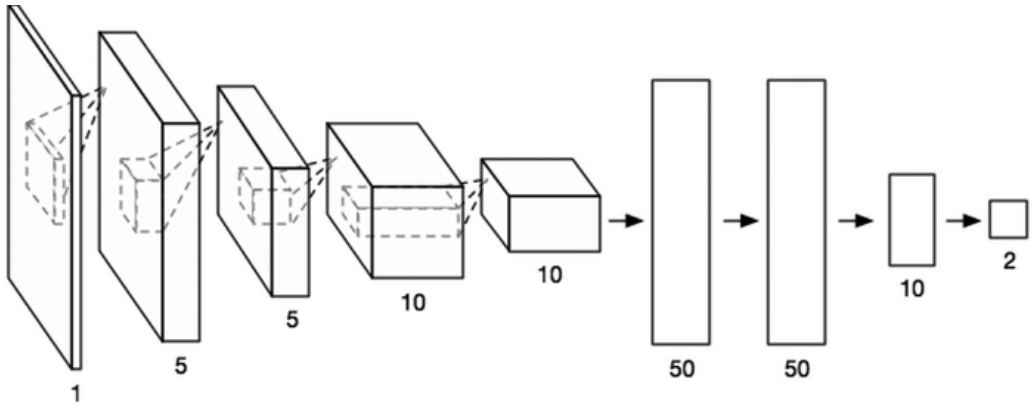


Figure 3.7: Convolutional Neural Network: Dimensionality reduction example for binary classification problem

Convolutional Neural Networks take advantage of the input being images and conform its architecture to arrange image data in 3 dimensions: width, height and depth. Generally the output consists in a reduction of the full image to a single vector of class scores, achieved by applying several different layers of transformation through differentiable functions.

It is important to discuss some typical types of layers usually comprised within a CNN:

- **Convolutional Layer:** Computes the convolution operation by performing a dot product between neuron weights connected to local regions of the input image, exploiting spatially local correlation. The parameters consist of a set of learnable filters with small receptive fields that extend through the full depth of the input image

- **ReLU Layer:** The Rectified Linear Unit Layer (ReLU) applies a non-linear elementwise activation function $f(x) = max(0,x)$ effectively removing negative values (i.e. situations where a neuron should not fire) by setting them to zero. ReLU is often preferred, in the context of machine learning aplications, to other common activation functions such as hyperbolic tangent or sigmoid functions because it trains the neural network several times faster than its counterparts without a significant penalty to generalization accuracy.

- **Pooling Layer:** Performs a downsizing operation along the spatial dimensions of input images (width, height), effectively reducing its volume. The role of a Pooling Layer is to reduce the resolution of the feature map while retaining features of the map required for classification through translational and rotational invariants. This property is especially important in Vision-based Robotic applications

Dropout and Normalization layers are also frequently incorporated in Convolutional Neural Networks to reduce overfitting and prevent internal covariance shifts within activation layers, respectively.

3.7.5 Recurrent Neural Networks (RNN)

Recurrent neural networks are a class of neural networks whereby connections between nodes form a computational graph along a time sequence, allowing it to capture temporal dynamics of video clips. Unlike previously explained Convolutional Networks, RNNs can store and use their internal states for capturing the information in the sequence of inputs itself, rather than the information in individual input images. The main principle behind RNNs is maintaining the memory of hidden states over time via feedback loops and modeling the dependencies between current and previous states.

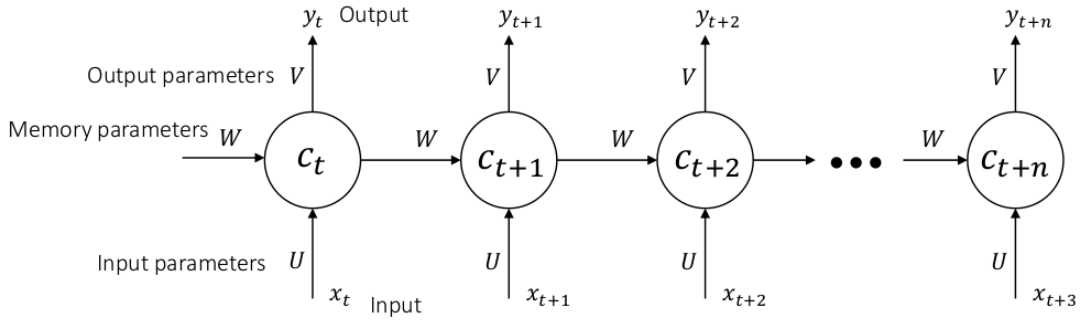


Figure 3.8: Recurrent Neural Network diagram

3.7.6 Long Short-Term Memory Units (LSTM)

The most relevant state-of-art RNN architecture worth mentioning in the scope of this thesis are Long Short-Term Memory Units (LSTM), a type of neural network first introduced by Hochreiter in 2007 [65]. A typical LSTM unit is comprised of a cell, an input gate, an output gate and a forget gate. The cell remembers values and keeps track of dependencies over random time intervals (there can be lags of unknown duration between important events in a temporal sequence); the input gate controls the flow of information into the cell; the forget gate controls the extent to which information is stored within the cell and the output gate determines when the values in the cell will be used to compute output activation of the LSTM unit.

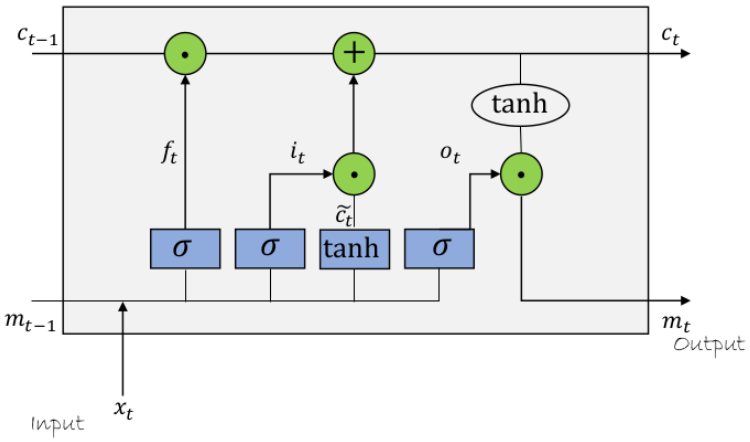


Figure 3.9: LSTM unit

3.8 Evaluation Metrics

In order to evaluate the results of the method implementations, it becomes necessary to compare between our pose estimates defined as a sequence **P1,..,Pn** $\in$ *SE(3)* and groundtruth data defined as **G1,..,Gn** $\in$ *SE(3)*. It is worth noting that the evaluation procedure works under the assumption that both pose sequence sets are time-synchronized and equally sampled, are both referenced to the same global frame and have the same length n.

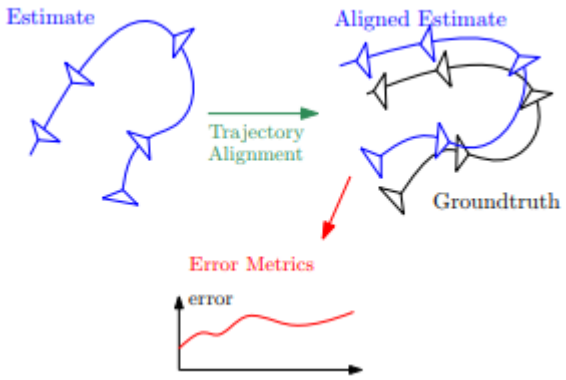


Figure 3.10: The process of quantitative trajectory evaluation. The trajectory estimate undergos a preprocessing alignment step before computing the estimation error. Image credits to [66]

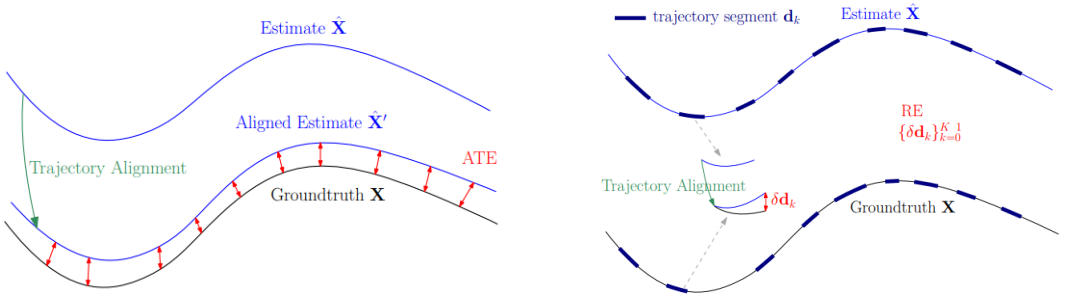
The accuracy of Visual Odometry methods is quantified through evaluating the estimated trajectory with respect to the groundtruth, which is a necessary definition for understanding and benchmarking different algorithms

In the remainder of this section, we present the evaluation metrics we used to assess method performance and accuracy. In particular, we report Absolute Trajectory Error (ATE) [54] as a test metric as well as RMSE of the global positions since Deep Learning methods usually take into account full sequence information. Here, a brief description of each metric is provided:

Relative Translation/Rotation RMSE: This metric takes the output of the model, reparameterizes it to position+quaternion format, and then takes the RMSE with the ground-truth relative-frame position+quaternion. Therefore this metric is the error in the estimated relative velocity/angular velocity between each image pair in the sequence. Since this metric only looks at relative transformations, it does not account for accumulating drift errors.

Global Translation/Rotation RMSE: This metric takes the output of the model, reparameterizes in position + quaternion format, but further integrates the relative transformations over time to produce the estimated global transformations of each pose. The RMSE between the estimated global SE(3) pose (position and attitude) and the groundtruth global-frame pose is then taken to produce the global translation or rotation error metric.

Absolute Trajectory Error (ATE): In our work, the ATE metric is the most useful metric for correctly evaluating method accuracy and performance. This is similar to the global RMSE metric, but it includes a preprocessing step described in [54]. This "aligning" preprocessing step seeks to align the estimated sequence with the ground-truth sequence by producing a scale, rotation and translation transformation that will be applied to every pose in the sequence. After aligning, the RMSE is taken between the aligned estimated global position and the groundtruth global frame position.



(a) ATE (b) RPE

Figure 3.11: Illustration of absolute trajectory error and relative pose error, credits to [66]

In addition, we are abiding by the Evo [67] software package nomenclature, as it provides a comprehensive toolkit for result evaluation. As so, it becomes relevant to further expand on the nomenclature:

Absolute pose error (APE) is a computation of the absolute trajectory error along the full trajectory sequence. Corresponding 3D translational vectors are directly compared between estimate and reference given a pose relation (i.e. temporal timestamps). This metric is useful to test the global consistency of the estimated trajectory and can be defined as

$$E_i = G_i^{-1} S P_i \tag{3.11}$$

where S corresponds to a rigid body transformation that is computed according to one of the preprocessing strategies that will be detailed later on in this section. The mean average error and standard deviation can then be computed along the entire sequence of data, as well as the RMSE.

Relative Pose Error (RPE) on the other hand, instead of directly comparing absolute poses, computes relative pose error in motions (i.e. "pose deltas"). This metric gives insights about the local accuracy of the method, and it corresponds to odometry estimates on Robotic applications. We are interested in evaluating 6-Dof pose estimates, but also separately look to isolated position and attitude errors.

$$E_i = (G_i^{-1} G_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}) \tag{3.12}$$

Then we can calculate the RSME in the entire sequence of data as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (E_i)^2}{N}} \tag{3.13}$$

Furthermore, we are going to utilize 3 different pre-processing evaluation condition scenarios:

- "Raw" comparison with no pre-processing
- Scale correction of pose estimates to match groundtruth data
- SE(3) Umeyama aligment [68]. This pre-processing step aligns the pose estimates with the groundtruth using a least-squares estimation of transformation parameters translation, rotation and scale (matrix S in equation 3.11)

Chapter 4

Deep Learning Approaches for Visual-based Robot Navigation

4.1 Introduction

This chapter denotes the scientific and applicational work performed during the course of the thesis. Namely, we will provide detailed explanations about the implementation of some of the most renown deep learning approaches for motion estimation, as well as evaluate and benchmark their performance in challenging underwater operational scenarios, when comparing with groundtruth information.

For this purpose, we first introduce a underwater visual dataset, that will be used to compare the methods results, explaining how it was constructed, as well as the image acquisition methodology, its reasoning and underlying assumptions.

Secondly, we tackle the robot relocalization task, evaluating two different state-of-the-art algorithms that allow for visual-based robotic localization, Posenet [41] and PoseLSTM [43]. The main purpose is to investigate whether these algorithms are able to perform robustly in more complex underwater scenarios.

Lastly, we turn our attention to the egomotion estimation task. Two landmark frameworks, SfM-Learner [36] and GeoNet [39] were implemented, so as to evaluate their performance on real robotic application mission scenario.

4.2 Underwater Visual Dataset

Deep learning methods usually require vast amounts of data in order to properly train its neural architectures. This is particularly true in robotic applications, since autonomous systems can operate in very complex environments, often under extreme conditions. As so, the availability of large scale datasets is crucial for further development of deep learning algorithms and its respective generalization ability, therefore improving upon its robustness when being deployed in full-scale large complex environments.

In the underwater context, there are not many publicly available large datasets and there is none widely regarded as a comprehensive benchmark for method evaluation. In the scope of this work, we also wanted to assess method performance using one of CRAS robotic solutions, namely the UNEXMIN UX-1 robot. With this in mind, we developed a Deep Visual Underwater dataset, an underwater focused dataset collected with the UX-1, tailored for visual odometry method implementation and evaluation, with which we pretend to assess performance of state-of-the-art Deep Learning architectures for VO estimation in different underwater scenarios (see Fig. 4.2.2). In Fig. 4.1, we can observe example images of our dataset sequences, that showcase the different environments included in our dataset.

In this section, we are discussing in detail the data acquisition process, specifically describing the UNEXMIN UX-1 robot and all the technology contained within it, while providing related remarks about the image acquisition methodology, specifically the camera setup,the reasoning and assumptions of the process.

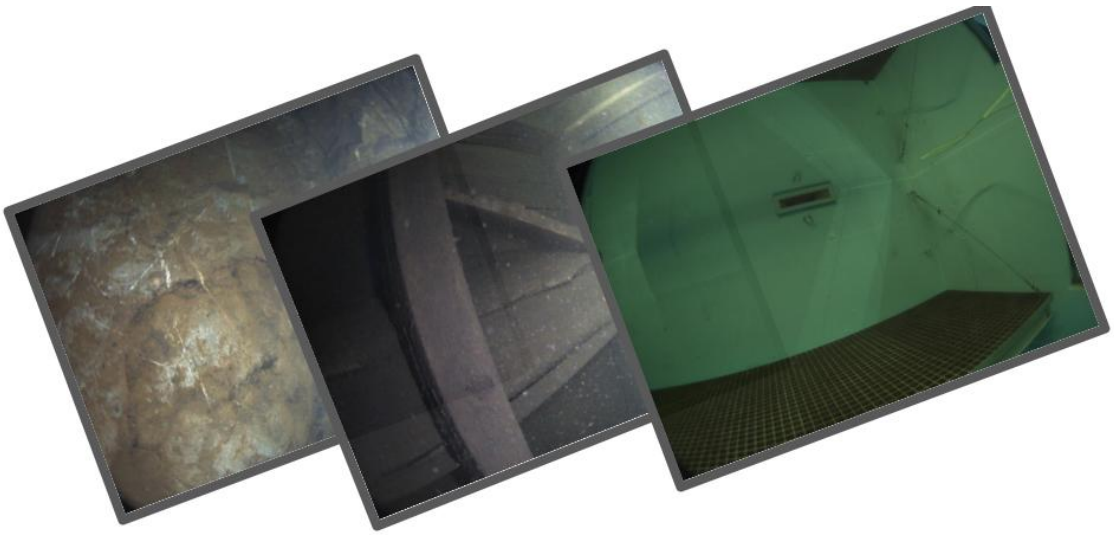


Figure 4.1: Dataset image examples

4.2.1 The Robotic Solution

In this work, though any of previously mentioned CRAS robots (Fig. 1.1) could be used, the focus of our analysis is the UX-1 robot (see Fig.4.2), one of three robots developed in the context of the UNEXMIN project [2].

The UNEXMIN project goal is to develop a fully autonomous multi-platform robotic exploration solution for the purpose of exploring and 3D mine mapping of flooded decommissioned mines that are otherwise inaccessible. It consists of three similar spherical shaped robots that carry different scientific instrumentation while sharing most of the software design specifications.

Figure 4.2: UNEXMIN UX-1 CAD design

The three UX-1 robots are equipped with full stack software for positioning, navigation, control and 3D mapping system. To fuel the algorithms performing all these tasks the robots are equipped with: an industrial grade Inertial Measurement Unit (IMU); a Doppler Velocity Logger (DVL); a Structured Light Systems (SLS) consisted of four LED's and laser projectors; one multispectral camera; a multibeam sonar; a Scanning sonar and five visible cameras (see Fig.4.3).

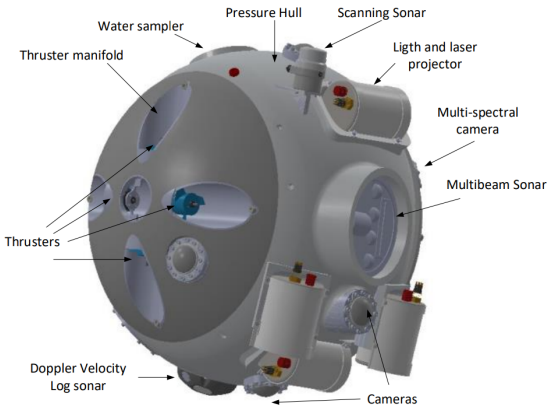


Figure 4.3: UNEXMIN UX-1 robot description

The information obtained by all these sensors is processed by the perception subsystems and the combination of this sensorial information feeds the navigation module where accurate and reliable vehicle navigation estimates are produced. This architecture allows the robot to accomplish the dual purpose of exploring and mapping the mine walls while accurately navigating autonomously.

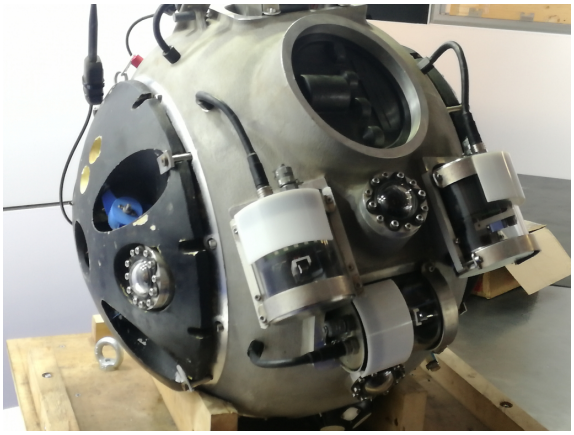


Figure 4.4: UNEXMIN UX-1 robot photograph

4.2.1.1 Robot Operating System

The UX-1 runs on ROS (Robot Operating System) [69], with a distributed architecture bounded by a common clock synchronization and timestamping. This means we can assure time synchronized data from multiple cameras and groundtruth robot global pose.

ROS is a distributed computing environment which may comprise multiple machines running multiple nodes; this is the case for the UX-1 which has multiple CPU's running multiple subsystems. Further expanding upon the nomenclature, ROS is based on a graph architecture with a centralized topology, where processing takes place in nodes that may receive, post sensorial information, exercise control, access state information, execute planning tasks or actuate in the environment. The key concepts of the computational graph in a ROS architecture are:

- **ROS Master**: The ROS Master provides a broker for exchange of information between all processes of the computational graph elements. It stores topics and service information for ROS nodes.
- **Nodes:** Nodes are processes that perform computation of subsystems in robotic applications. A ROS system usually comprises many different nodes that control every aspect of the robotic system from sensorial information processing to actuation in the environment
- **Messages:** Messages define a data structure, comprising typed fields that are used as a way to communicate between nodes. Standardized message fields ease the burden of cross implementations, as interfaces can be easily interpreted and understood.

- **Topics:** Topics identify the content of the message. Topics work essentially as information channels with publish/subscribe semantics. Different nodes communicate by sending messages by publishing them on a given topic.
- **ROS Bags:** Bags are a format for saving and playing back ROS message data (topics). Bags store all the information flowing in a ROS system at the time of the recording.

The ability to record all data published in a ROS system for later processing was crucial for our work, as we make use of this record function to acquire our data, saving it to ROS bags that we can process to build our annotated visual dataset.

4.2.1.2 Data acquisition methodology

As previously mentioned, the dataset was constructed with data acquired with the UX-1 robot. This robotic solution is equipped with a plethora of different sensors, including 5 cameras. In this work, and especially since the UX-1 does not have a great overlay of camera fields-of-view, we are focusing on monocular visual methods, and as so, we choose to analyze the left and right camera systems, with the goal of estimating robot pose in the central reference frame (i.e. pose estimates from both camera systems have to be later transformed to the robot body reference frame).

Groundtruth data is generated by the navigation module of the UX-1 software, a filtered calibration of sensor fusion from the multiple sensor sources presented above, progressively refined through multiple operation missions in complex settings and extremely challenging operational conditions. In the scope of this work, we are working with the underlying assumption that this navigation data corresponds exactly to the real robot pose, which is not easily verifiable in operational mission scenarios. However, it can be asserted, with relative confidence, that this data represents a close approximation of the real robot position and can, therefore, be used as groundtruth for our use case. The groundtruth data file consists of a *.txt* file where each line contains 8 scalars, representing a timestamp and 6-DoF poses with a 3D translation vector and an orientation quaternion.

4.2.1.3 Data preprocessing

In order to process the raw data contained in the above mentioned ROS bags, we developed a ROS package with the purpose of automating time-synchronized data extraction. Since visual information is naturally published at a lower sample rate than navigation information, a timestamp matching based on a basic interpolation process was implemented, where the goal was to obtain reliable data with precise synchronization between the images and the groundtruth pose data.

Algorithm 4 Time-synchronization of visual data and groundtruth pose

After ROS subscriber and callback implementation,

- Subscribe to camera topics and navigation pose data.
- Save higher frequency navigation data to queue buffer, with associated timestamp t_{nav}

For every image callback,

- Extract image timestamp t_i
- Find correspondences within a confidence interval $\|t_{nav} - t_i\| < threshold$
  If correspondence is found :
  1. Extract images and corresponding pose data to dataset directory
  2. Append timestamp and camera intrinsic calibration files to directory

  Else :
  1. Drop image and proceed to next image callback

This processing step becomes necessary because whilst recording the ROS bag, the camera topic is sometimes delayed in relation to the navigation data, due to the mismatch in size of the image data being written to disk when compared to a simple pose message.

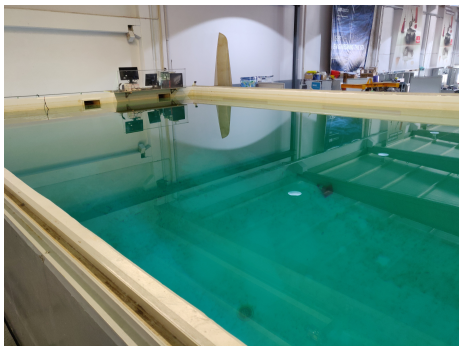
In addition, our preprocessing package appends to the dataset directory 2 additional files: one containing synthetic timestamps generated starting the clock at the beginning of the simulation and a file containing the respective camera intrinsic calibration parameters.

Later in this work, the algorithm presented above was further extended so as to account for the possibility of extracting synchronized images from multiple cameras, making it also suitable for the stereo use case.

4.2.2 Scenarios

For the purpose of constructing a complete and thorough dataset, we utilize three different application scenarios, which all pose different types of problems to visual-based methods:

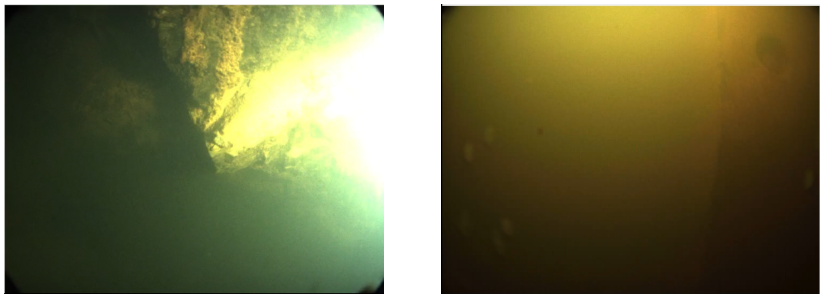
- **The CRAS pool** sequence depicts a fully known environment, ideal for calibrating some aspects of visual-based navigation, since all navigation information is fully verifiable.



- **The Urgeiriça uranium mine** is a decommissioned flooded mine in Viseu, Portugal. It is mostly composed of vertical shafts that lead to 15-30m wide galleries. It is a real operational mission scenario for the UX-1, which was tasked with exploring and mapping the mine.

- **The Ecton copper mine** is also a flooded mine, situated in the United Kingdom, with particular interest for visual odometry estimation, due to the high quality of imagery it can provide. Water clarity and reliable illumination conditions make this image sequence a really interesting operational mission scenario.

4.2.3 Environment constraints



(a) Artificial lighting effects (b) Water turbidity

Figure 4.5: Environmental constraints in underwater VO: In the left image it is possible to observe artificial lighting effects on image quality. In the right image, it is visible the water turbidity and the effect of suspended particles may cause on underwater imagery. Both images were acquired in operational scenarios by the EVA robot [1]

.

Acquiring underwater imagery is a very challenging task. Setting aside mechanical constraints such as camera waterproofing, the necessity arises for the application of artificial lighting in predominantly dark scenes. This hinders the quality of the acquired images and may introduced undesired shadows in the image acquisition process. To summarize, typical underwater computer vision systems face the following issues:

- **Lack of appropriate lighting conditions:** Water absorbs light in pretty much all wavelengths, generally attenuating less green and blue wavelengths. A possible solution for this problem, as mentioned previously, is the use of artificial lighting. This solution generally produces brighter and higher contrast images, but usually struggles to capture information around the direct light incision region and introduce problems such as low contrast and non-uniform illumination.

- **Backscattering:** Diffuse reflection of light rays that causes image blur. Light coming from ambient illumination sources is scattered towards the camera through suspended particles.

- **Lack of image texture:** Textures significantly help in the identification of given features of the image. Often, underwater imagery present no distinguishable textures from which observer relative displacement can be calculated or scene depth can accurately be calculated.

- **Vignetting Effect:** Reduction of an image's brightness or saturation toward the periphery compared to the image center.

Underwater computer vision applications usually involve image restoration and enhancement steps and further color correction in order to produce high quality imagery for post-processing.

4.3 Relocalization and Global Pose Estimation

Posenet [41] was the first instance of a deep convolutional network used for regressing 6-DoF camera pose from visual input. The algorithm can be described in 3 steps:

Architecture: This supervised learning method was designed based on GoogLeNet [70], a state of the art deep neural network architecture tailored for classification tasks. The network was altered through replacing softmax classifiers with pose regressors within the final fully-connected layers. 6-DoF pose is represented with a 7 dimensional vector representing 3D position and an orientation quaternion.

Transfer Learning: The feature extraction process is pre-trained on large scale image classification datasets in order for pose regression to work off of comparatively small amounts of data, leveraging powerful representations learned on large scale image datasets. In our work, we used pre-trained weights on *Places* [71] as a starting point for feature extraction, later context finetuning the network to the underwater environment by further training with our dataset training sequences.

Loss Function The chosen loss function represents a weighted Euclidean distance loss between pose estimates and groundtruth pose labels:

$$loss = \|\hat{x} - x\|_2 + \beta \left\| \hat{q} - \frac{q}{\|q\|} \right\|_2 \tag{4.1}$$

where x represents the position vector, q represents rotations on the unit sphere in quaternion space and β is a tunable parameter used for penalizing equally errors in position and orientation. β was empirically tuned to the 100 to 500 range, which is in agreement with the author's findings for confined environments, where position errors tend to be small .

We also decided to evaluate PoseLSTM [43], a similar framework whereby accuracy was reported to improve through the added use of LSTM units for dimensionality reduction and structured feature correlation.

4.3.1 Results on Deep Visual Relocalization

We tested both methods on one of the author's proposed image sequence and afterwards on our own dataset, in order to to assess relocalization performance on the underwater context.

The experiments were conducted using Pytorch [72] implementations of both algorithms, as well as a custom developed Keras [73] Posenet implementation. Reported results comprise the best results we were able to obtain, empirically tuning the hyper-parameter β and otherwise adopting the paper's learning scheme.

Table 4.1: Average Error Results of Deep Visual Relocalization algorithms

Method	Scenario	Beta	Avg. Error
PoseNet	King's College	100	1.02m 4.76^o
		200	1.13m 4.60^o
		500	1.11m 4.46^o
	CRAS pool	100	0.14m 1.68^o
		200	0.15m 1.62^o
		500	0.16m 1.50^o
	Urgeiriça Mine	100	0.33m 5.84^o
		200	0.33m 3.94^o
		500	0.30m 2.31^o
PoseLSTM	King's College	100	0.92m 5.85^o
		200	0.90m 4.84^o
		500	0.93m 4.71^o
	CRAS pool	100	0.10m 2.04^o
		200	0.11m 1.59^o
		500	0.10m 2.38^o
	Urgeiriça Mine	100	0.32m 3.69^o
		200	0.33m 2.59^o
		500	0.34m 3.27^o

4.3.2 Generalization

It is worth noting that, similarly to the author's approach, these results pertain to relocalization performance within a chosen dataset sequence, thus lacking generalization robustness assessment. Intuitively, it is easy to understand that the usefulness of relocalization algorithms is closely tied to its ability to perform under a wide variety of complex situations, thus method generalization performance is key.

So as to evaluate the inference process generalization ability, different sets of full partitions of data were fed to the network, with a shared global reference frame yet comprising different sets of lighting conditions and variations in robot trajectory and velocity.

Table 4.2: Average Error Assessment of generalization ability of Deep Visual Relocalization algorithms

	β	CRAS Pool	Urgeiriça Mine
PoseNet	100	2.12m 13.76º	28.77m 77.49º
	200	2.13m 14.34º	28.08m 88.60º
	500	2.16m 15.04º	28.78m 88.70º
PoseLSTM	100	2.09m 14.69º	28.05m 77.10º
	200	2.07m 13.12º	28.10m 76.80º
	500	1.95m 14.29º	28.12m 76.68º

As it can be observed in Table 4.2, the generalization ability and general robustness of both methods to changes in the environment had somewhat mixed results.

In the small scale CRAS pool setting, the average errors in trajectory are still within a reasonable range however, orientation perception performance falls off significantly. Despite of the shortcomings, the results on localization still provide a fairly good picture of the robot trajectory (see Fig. 4.6), notwithstanding the very noticeable presence of outlier values.

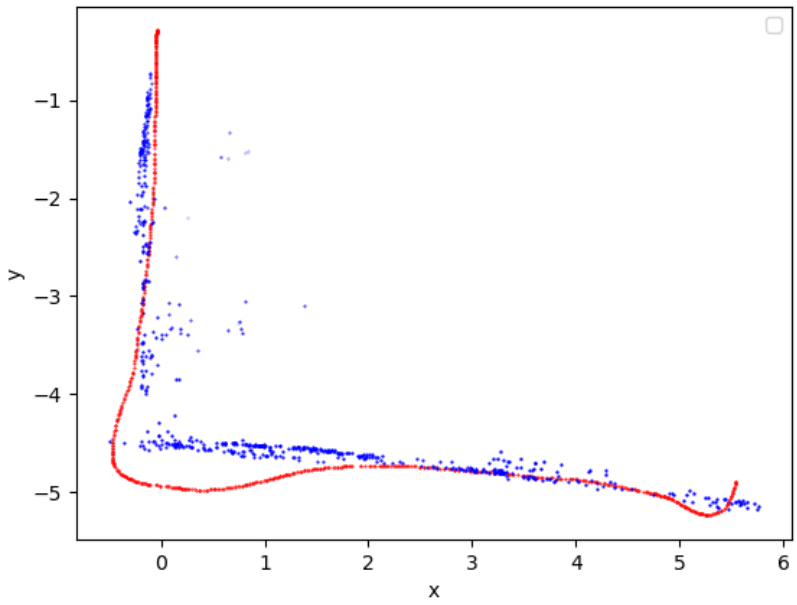


Figure 4.6: CRAS pool relocalization performance: In red, we can observe the robot groundtruth trajectory and in blue the predicted 2D coordinates. Note that position estimates are estimated as 3D translation vectors, but the z-dimension was omitted in this use case for the sake of representation (i.e. movement was only 2-dimensional)

Results on operational mission scenarios, on the other hand, present a somewhat lackluster performance in generalization ability, with the model not being able to successfully relocalize. However, the unfavorable conditions of the data must be taken into account: the homogeneity and overall textureless of the mine walls, coupled with uneven illumination conditions, present a uniquely challenging and complex scenario for visual based methods.

4.4 Egomotion Estimation

In the scope of underwater robotics research and development, and specifically in the context of our work, the most interesting application we are interested in exploring are unsupervised Deep Learning frameworks for egomotion estimation.

For the purpose of estimating motion dynamics, we are turning our attention to two similar state-of-the-art deep convolutional visual frameworks: SfmLearner [36] and GeoNet [39]. Though both frameworks also estimate monocular depth (and optical flow in the case of GeoNet), we are only focusing on camera motion estimation CNN's.

4.4.1 SfMLearner

SfMLearner is of an unsupervised learning pipeline for depth and egomotion estimation. The unsupervised objective is fulfilled based on the following intuition: given knowledge of camera self-motion within a sequence of images and the depth of every pixel in those images, we can gain an unsupervised target by performing view synthesis.

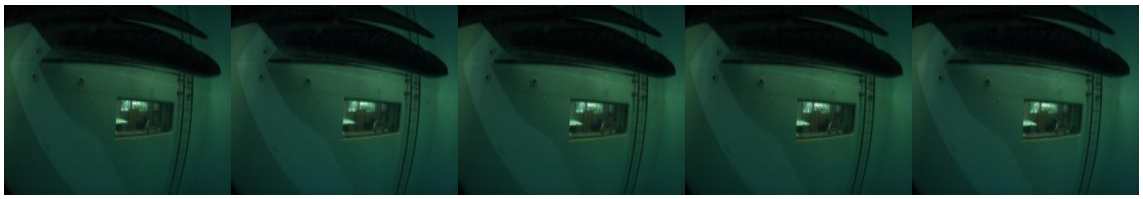


Figure 4.7: CRAS pool 5-sequence length snippet:

As mentioned above, we are interested in evaluating Zhou's PoseNet, the SfMLearner framework component responsible for regressing 6-DoF pose estimates. The PoseNet architecture is essentially a temporal convolutional network which processes a sequence of of n images by predicting relative transformation from the center image of the sequence (the image at the central position of the snippet, as shown in Fig. 4.7) to the other images in the sequence, outputting a *n-1* transformation vector composed of a 3D translation vector and a Euler angle orientation vector for each transformation.

The network itself is a convolutional regressor model with seven convolutional layers with stride-2 followed by ReLU activations, leading to a final linear convolution that outputs the aforementioned *6 x (n-1)*- dimensional channels. On top on this network, an "explainability" mask is used to downweight the loss on image patches undergoing motion external to the camera's motion (e.g. a car or pedestrian moving in the frame).

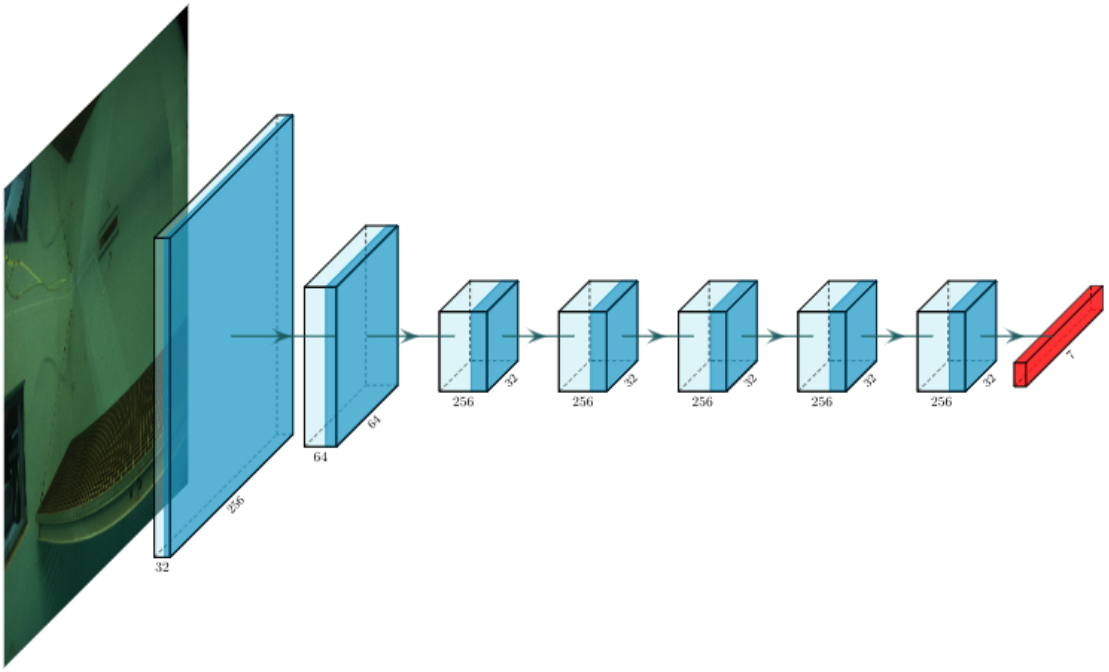


Figure 4.8: Representation of the SfMlearner PoseNet, the framework component responsible for regressing 6-DoF pose estimates. It consists of 7 blocks of convolutional layers followed by ReLU activations, outputting a 6-dimensional vector that comprises a 3D translation vector and euler angles orientation representation.

4.4.1.1 Procedure and Hyperparameter Details

We began by writing a custom dataset loader that fed the visual and groudtruth data to the framework in the same way the author does for the KITTI dataset, opting as well for a 5-snippet sequence length. Calibrated camera intrinsics and timestamp files were also introduced, constructing an input directory consisting of 416x128 sequences of 5 stacked consecutive images, intrinsic calibration parameters and groundtruth pose for training data.

We train all models using the Adam [74] optimizer, using a small hyper-parameter search over learning rates [0.0005,0.001,0.005] to determine a suitable learning rate that best fit the data. We found early that rotation errors were introducing large global trajectory errors due to unaligning the pose sequence with the groundtruth thus accumulating significant drift, and so we decided to penalize heavier errors in rotation.

Training took on average 250K epochs on our dataset to converge, with tensorflow[75] running on a CUDA enabled Nvidia GTX 1080.

In addition, a post processing step was implemented in order to recover the full trajectory from the 5-snippet predicts, so as to plot the results and analyze the concatenated global trajectory errors. As the network was trained with the camera reference frame, the results also underwent through the reference frame transformation mentioned in section 4.2.1, from camera to body reference frame.

4.4.2 GeoNet

GeoNet [39] is a jointly trained end-to-end unsupervised learning framework for monocular depth, optical flow and egomotion estimation. Specifically, this framework focuses on extracting geometric relationships in the input data by separately considering static and dynamic elements in the scene. Significant performance gains have been reported, mostly due to increased robustness towards texture ambiguity and occlusions in the scene.

The framework is composed of two stages: the Rigid Structure Reconstructor and the Non-rigid Motion Localizer. The first stage is tasked with understanding the scene layout and structure and it consists of two sub-networks, i.e. the DepthNet and the PoseNet. The second stage concerns itself with dynamic objects in the scene and it utilised for the purpose of refining imperfect results from the first stage due to motion external to the camera motion, as well as help deal with high pixel saturation and extreme lighting conditions.

Similarly to SfMlearner, view synthesis at different stages works as a synthetic supervision for the unsupervised learning architecture, with image appearance similarity enforcing geometric and photometric consistency within the loss function.

The most relevant part of the framework in the scope of our work is the Pose Net, which consists of 7 convolutional layers followed by batch normalization and Relu activation (see Fig. 4.9). The prediction layers outputs the 6-DoF camera poses, i.e. translational vectors and orientation Euler angles.

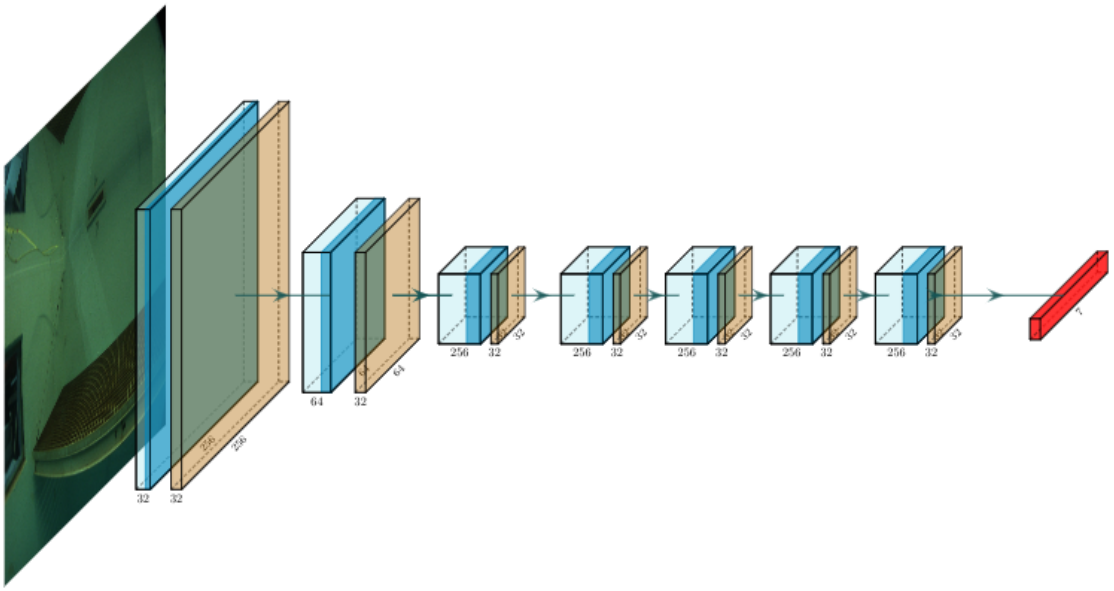


Figure 4.9: Representation of the GeoNet PoseNet, the framework component responsible for regressing 6-DoF pose estimates. It consists of 7 blocks of convolutional layers followed by ReLU activations and additional batch normalization layers, outputting a 6-dimensional vector that comprises a 3D translation vector and euler angles orientation representation.

4.4.2.1 Procedure and Hyperparameter Details

GeoNet's input data shares the same preprocessing steps mentioned in the scope of the SfMlearner experiment. A similar sequence lenght of 5 image snippets in conjunction with camera intrinsics was fed to the network, training only the Rigid Structure Reconstructor half of the framework.

The chosen optimizer was Adam, and a learning rate sweep was performed around the intervals [0.0001,0.0002,0.0005,0.001] where $\beta 1 = 0.9$ and $\beta 2 = 0.999$. Due to high number of trainable parameters in the network, a reduced mini-batch size was utilized and training converged on average after 200K iterations on the training data.

The training setup was also the same, running the architecture on top of tensorflow, with an Nvidia GTX 1080.

Global trajectory was recovered from 5-frame snippets odometry predicts and transformed to the robot body central reference frame, in an analogous way to how we did with SfMLearner.

4.4.3 Results

Results on our dataset are presented in two different forms. First, we evaluate in a similar fashion to how both authors presented them, by computing estimate errors within the previously mentioned 5-frame sequences, with scale correction optimization and alignment with groundtruth data, so as to resolve scale ambiguity. In addition, we decided to evaluate full sequence pose estimates, through concatenating camera position inferences to a global trajectory and computing both absolute position errors and relative pose deltas. The same scale correction and groundtruth alignment process that both authors use on their publications was also utilized.

In order to contextualize the reader about state-of-the-art method performance on a common benchmark dataset, we begin by presenting the results of motion estimation on the KITTI dataset, that were able to be accurately reproduced. Additional information, not present on the papers, about the full concatenated trajectory is introduced and presented before we proceed to access the performance of the same algorithms on the underwater context.

Table 4.3: Absolute Trajectory Error (ATE) method evaluation: The presented results comprise the best results we were able to obtain for each architecture in each dataset sequence, following the author's evaluation procedure within 5-frame snippets

	KITTI seq 09	KITTI seq 10	CRAS Pool	Urgeiriça Mine
SfMLearner	0.016 ± 0.009	0.013 ± 0.009	0.016 ± 0.006	0.028 ± 0.086
GeoNet	0.012 ± 0.007	0.012 ± 0.009	0.012 ± 0.006	0.026 ± 0.081

In table 4.3, we report the results of ATE method evaluation, following the same evaluation procedure conducted by the respective authors, whereby relative motion is evaluated only locally within the 5-frame sequence length snippets. ATE is computed on 5-frame snippets and averaged over the full sequence. Right away it is possible to observe that we were only able to reproduce same magnitude performance in the CRAS pool sequence, with comparatively poorer accuracy in the more complex Urgeiriça mine sequence.

In the remaining of this chapter, we will present and discuss the results considering the full concatenated trajectory, thereby escaping the snippet representation and recomputing errors with respect to translation for all sequence trajectories under analysis.

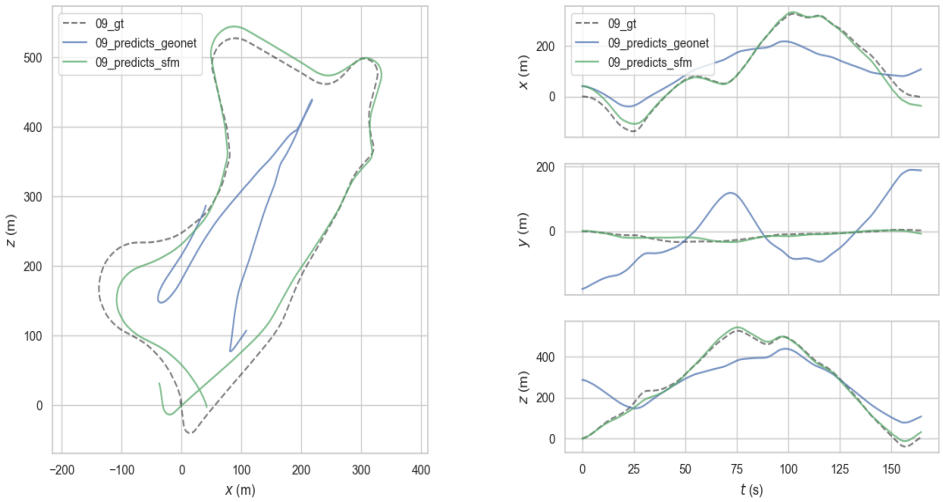


Figure 4.10: Results for KITTI sequence 09: on the left image we can see the computed trajectory estimates against groundtruth data and on the right hand side the error decoupled by translation axis

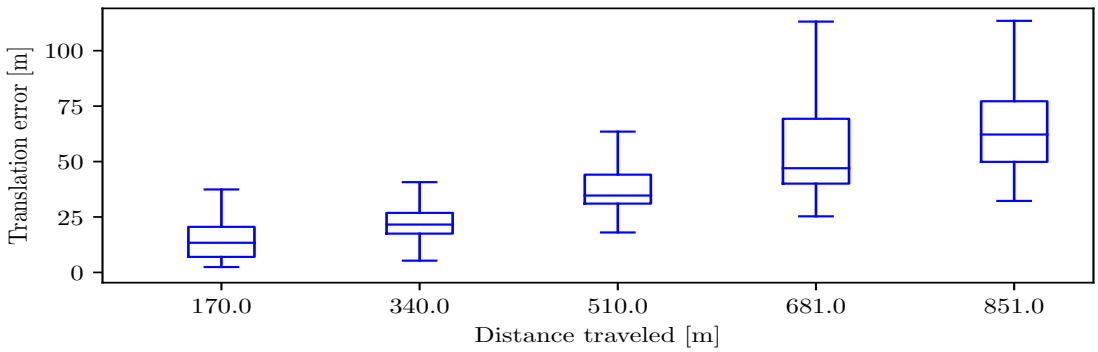


Figure 4.11: Translation Error with respect to distance traveled from SfMlearner pose estimates in KITTI sequence 09

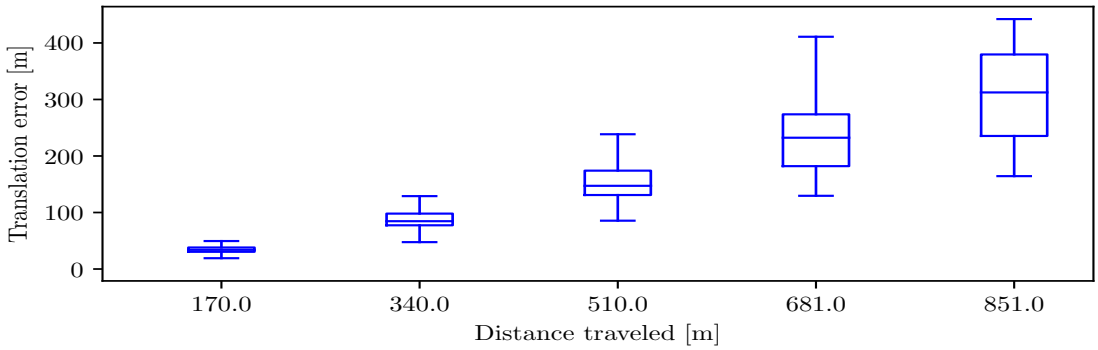


Figure 4.12: Translation Error with respect to distance traveled from GeoNet pose estimates in KITTI sequence 09

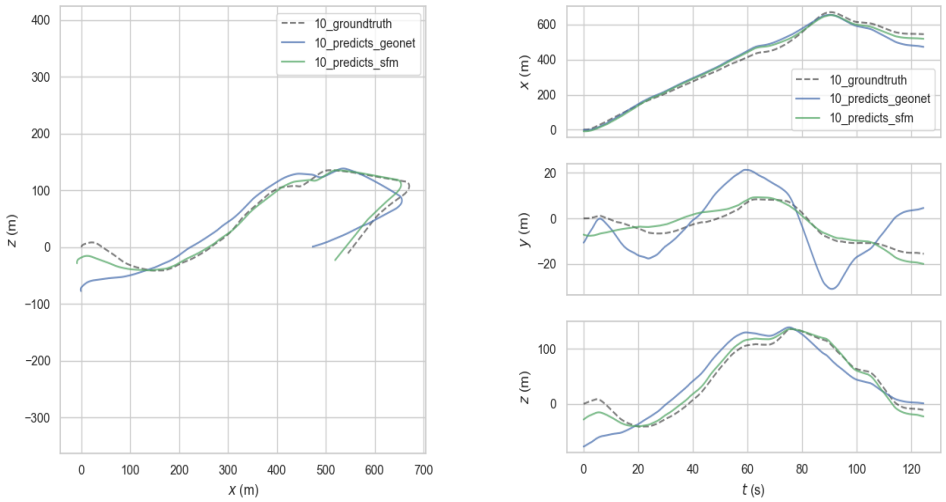


Figure 4.13: Results for KITTI sequence 10: on the left image we can see the computed trajectory estimates against groundtruth data and on the right hand side the error decoupled by translation axis

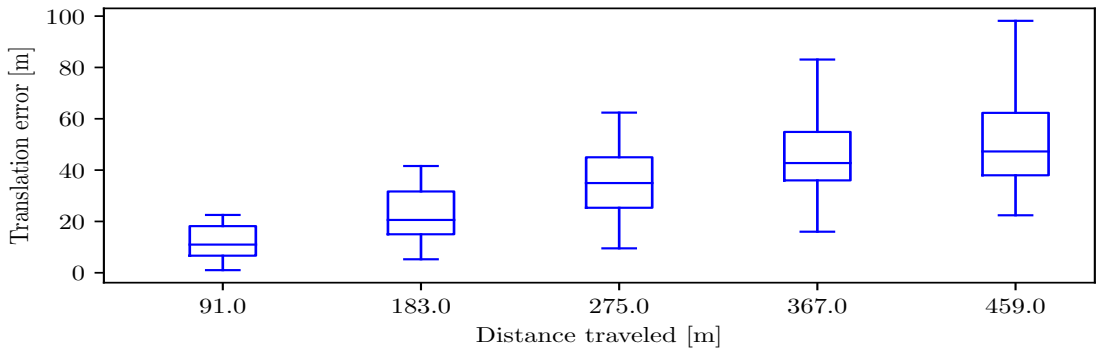


Figure 4.14: Translation Error with respect to distance traveled from SfMlearner pose estimates in KITTI sequence 10

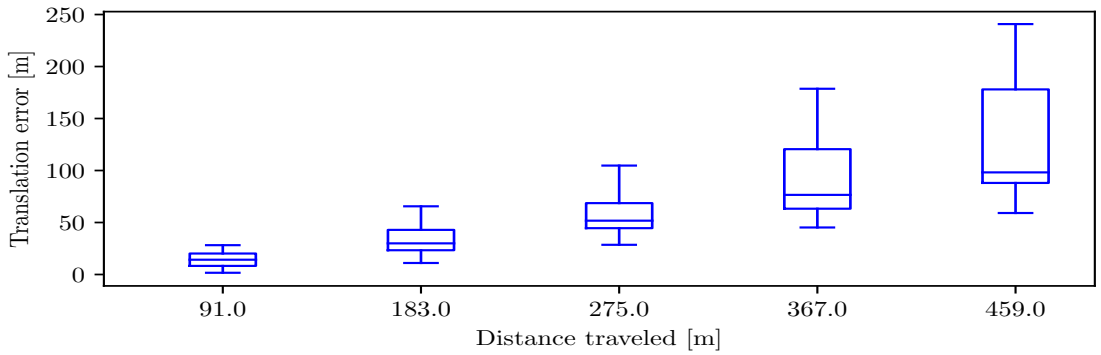


Figure 4.15: Translation Error with respect to distance traveled from GeoNet pose estimates in KITTI sequence 10

Table 4.4: Absolute Pose Error (APE) w.r.t translation: Compilation of best obtained results on full concatenated trajectory in the global reference frame, comprising both algorithms and all training procedures.

	Absolute Pose Error (APE)					
	"raw" comparison		scale-corrected		SIM(3) Umeyama aligment	
	Avg.Error	RMSE (m)	Avg.Error	RMSE (m)	Avg.Error	RMSE (m)
CRAS POOL	3.301±2.049	3.996	2.755±1.573	3.049	0.731±0.440	0.905
Urgeiriça Mine	52.709±1.199	52.461	20.354±3.366	19.129	0.721±0.584	1.077

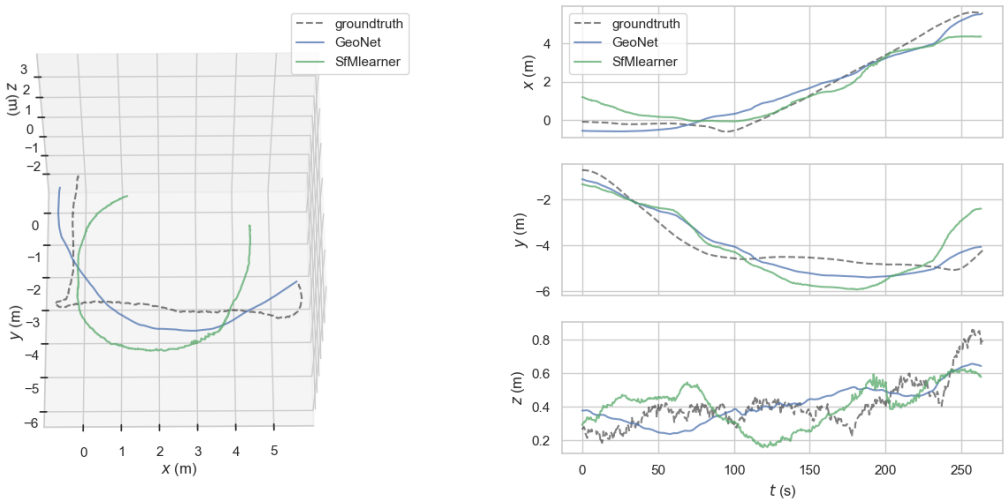


Figure 4.16: Results for CRAS pool sequence: on the left image we can see the computed trajectory estimates against groundtruth data and on the right hand side the error decoupled by translation axis

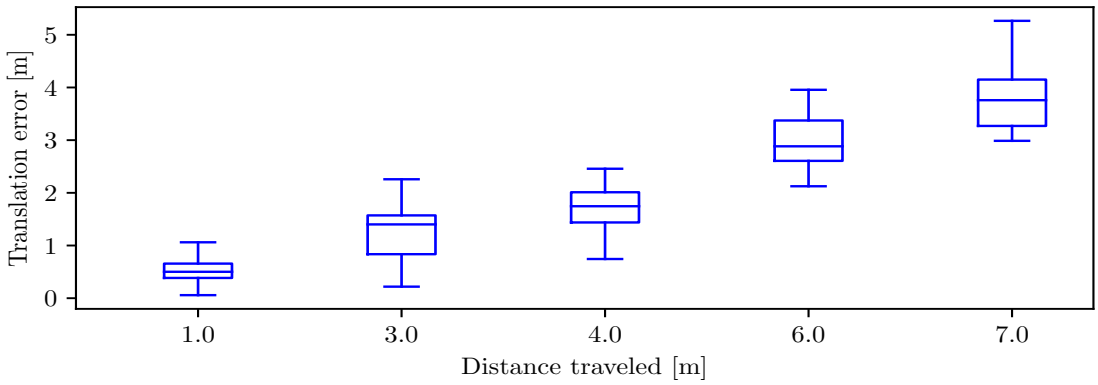


Figure 4.17: Translation Error with respect to distance traveled from SfMlearner pose estimates in our CRAS pool sequence

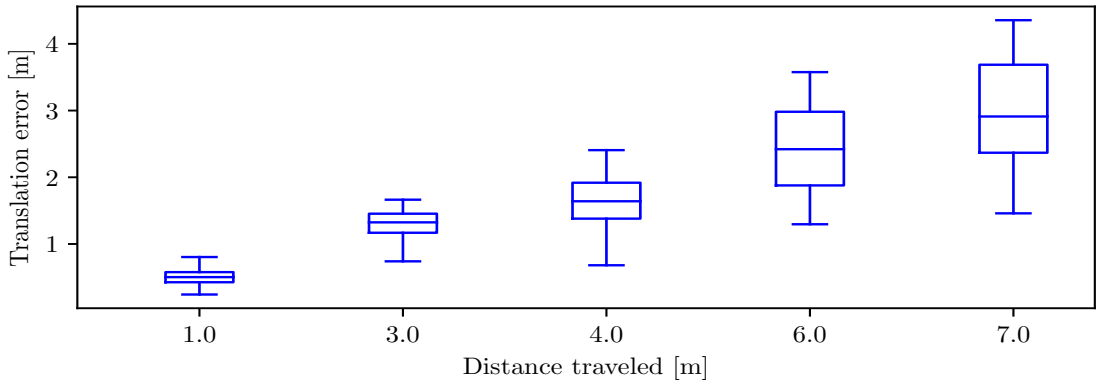


Figure 4.18: Translation Error with respect to distance traveled from GeoNet pose estimates in our CRAS pool sequence

Table 4.5: Relative Pose Error (RPE): Evaluation of relative pose error provides insight about the local accuracy, i.e. the visual odometry drift of pose estimates

		Relative Pose Error (RPE)					
		6-Dof Pose		Position		Attitude	
		Avg.Error	RMSE	Avg.Error	RMSE	Avg.Error	RMSE
SfMLearner	CRAS Pool	0.034±0.027	0.048	0.022±0.017	0.031	0.022±0.024	0.037
	Urgeiriça Mine	0.028±0.254	0.265	0.014±0.06	0.142	0.020±0.216	0.224
GeoNet	CRAS Pool	0.027±0.094	0.157	0.017±0.092	0.154	0.013±0.020	0.030
	Urgeiriça Mine	0.023±0.218	0.261	0.010±0.048	0.139	0.019±0.214	0.221

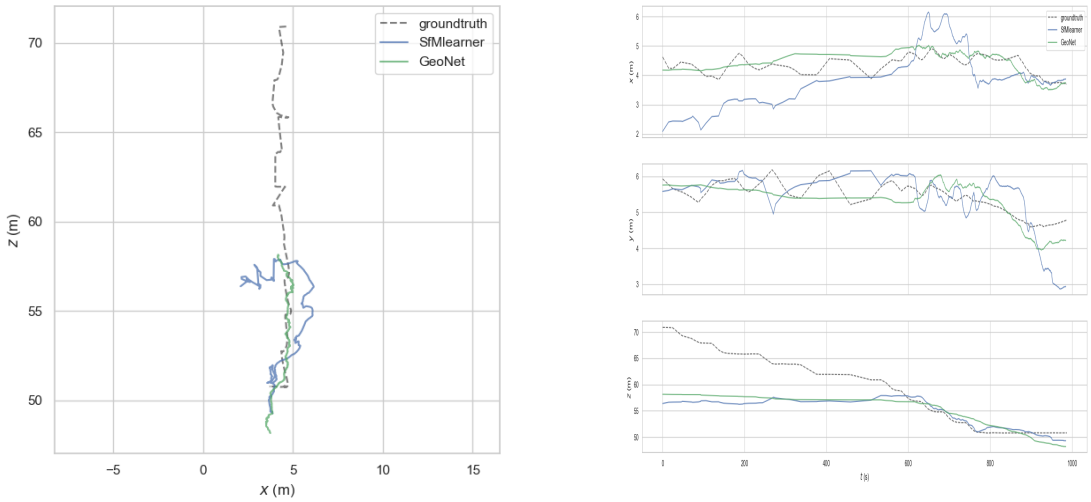


Figure 4.19: Results for Urgeiriça mine sequence: on the left image we can see the computed trajectory estimates against groundtruth data and on the right hand side the error decoupled by translation axis

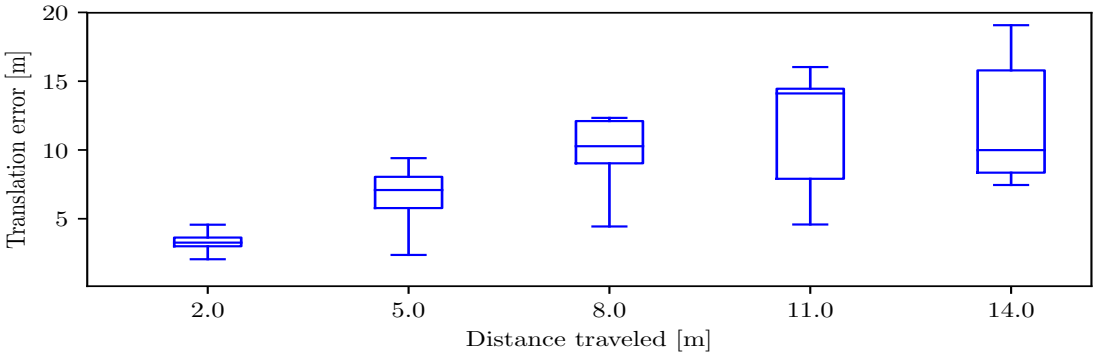


Figure 4.20: Translation Error with respect to distance traveled from SfMlearner pose estimates in our Urgeiriça mine test sequence

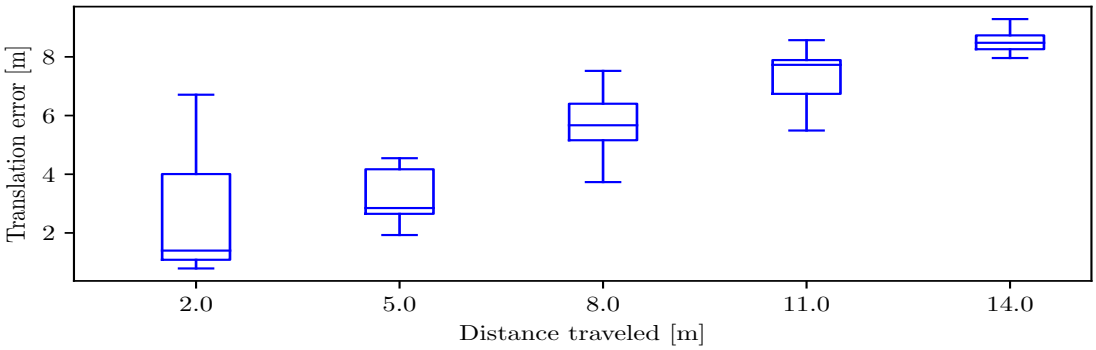


Figure 4.21: Translation Error with respect to distance traveled from GeoNet pose estimates in our Urgeiriça mine test sequence

4.5 Summary

This chapter analyzed the performance of Deep Learning methods on robotic relocalization and egomotion estimation tasks, with particular emphasis on the underwater context. Localization is one of the most fundamental competencies required by an autonomous robot as the knowledge of a robot position in the environment can be a prerequisite for SLAM systems or an adequate strategy for solving the robot "kidnap" problem, serving essentially as a precursor to the decision making process about future actions.

Underwater visual-based navigation poses different challenges compared to the common benchmark urban scenarios, thus further highlighting the necessity to assess the performance of such methods in the underwater context. The underwater visual dataset developed in this work encompasses both a controlled structured environment and operational mission scenario data, therefore allowing for an in depth study of deep learning method performance and accuracy

The study of deep visual relocalization architectures yielded two fundamental conclusions: deep learning methods can achieve satisfactory accuracy and performance in complex scenarios but the generalization ability of the methods remains somewhat lackluster, not showing great robustness to variations in lighting conditions and variations in robot trajectory and velocity. However, it can be assumed that deep visual relocalization algorithms have the potential to be applied in real robotic solutions equipped with suitable GPU hardware, though real time testing of the architectures is still required.

The performance of state-of-the-art deep learning methods for egomotion estimation can be analyzed through different perspectives, leading to the following conclusions:

- First of all, as it can be observed in table 4.3, we were able to produce similar results to those presented in the literature only for our CRAS pool sequence. It is still a good indication that it was possible to achieve such results in the underwater context, however, it is important to note that it was only true for our fully known structured environment. Real mission operational scenarios like the Urgeiriça mine sequence pose greater challenges to visual-based motion estimation algorithms and that is reflected on higher magnitude error rates.

- Secondly, it is possible to observe that both networks performs fairly better at regressing translational displacement than rotational movement. Rotation, and in particular pure rotations, are not handled well in any of the studied methods.

- In accordance to the expectations, and in agreement with both authors result presentation, pose estimates only present persuasive results with a post-processing step. The need for scale correction is a consequence of the use of monocular camera setups, but some type of groundtruth alignment algorithm is also required.

- Though relative motion estimates seem at first glance to show potential due to small average error rates, their concatenation onto the full trajectory reveals that the drift accumulation results in poor trajectory shape mimicking. In conclusion, there is still room for improvement when it concerns to global pose estimation derived from unsupervised egomotion estimation frameworks.
- Inference time may present a bottleneck for real-time implementation, since egomotion estimation frameworks are somewhat computationally heavy. However, it is reasonable to assume that the trained models could be used in conjunction with inference optimization software like TensorRT, allowing for the lower latency and higher throughput required by real-time applications.

Chapter 5

Global Trajectory Optimization

5.1 Introduction

In the previous chapter, results of state-of-the-art deep learning methods for egomotion estimation on our dataset were presented. As discussed in the previous chapter summary, the performance and accuracy of such methods in underwater deep mine environment was not at all satisfactory since there was significant accumulated drift in the global trajectory. This would severely hinder the prospects of deploying such architectures in real applications, as robot localization would not be reliable a short time after initialization.

With this in mind, two possible novel solutions for improving upon global 6-DoF global pose estimates generated by egomotion estimation methods were conceived and implemented. In this chapter, we will discuss and present the results of both architectures and access their impact on correcting global trajectory estimates. Note that the purpose of this methods is not to work independently of the previously studied algorithms, rather working subsequently to the egomotion estimation frameworks, effectively correcting the introduced drift on the trajectory estimates. The ultimate goal would be to incorporate this developed subsystems in an end-to-end fashion with the state-of-the-art solutions, helping to improve upon accuracy and performance of such methods.

5.2 Visual Inertial Fusion Network

Regardless of the algorithm, traditional monocular VO solutions are unable to observe the scale of the scene and are subject to scale drift and scale ambiguity. This is not different for Deep Neural architectures, as reported in the previously studied frameworks. The most common approach for pose optimization in the literature is to fuse visual and inertial data as a way to enforce global scale consistency with respect to the groundtruth data and therefore it would make sense to investigate analogous deep learning approaches to perform this task.

In this work, we propose a Recurrent Neural Network architecture anchored in a supervised learning scheme whereby we use filtered IMU readings as a supervision for 6-DoF pose estimate optmization.

The **input space** of this network are the concatenated egomotion predictions of both SfmLearner and GeoNet, i.e. global trajectory estimates in the robot central body frame. For this purpose, and due to deep learning architectures requiring large amounts of data to converge to a robust model, we had to run multiple predictions from both frameworks so as to synthetize a dataframe dataset.

The **network** itself consists of stacked LSTM units working with progressively smaller time step lags leading to a multilayer perceptron that regresses the optimized trajectory estimate. The goal is to process the data as a sequence-to-sequence problem, optimizing the input trajectory estimates to a more globally consistent trajectory.

The **fundamental assumption** driving this architecture is that the output space of the optimized trajectory estimate lie in a manifold much smaller than 6-DoF space. Implicitly constraining the output prediction space to a minimization of the mean square error between visual and inertial data helps to avoid the curse of dimensionality.

For **loss function** design , the intuition was that we needed to make use of the quaternion parametrization to penalize rotation errors in a meaningful way. In this light, we decoupled the translation and rotation components and formulated a loss function that takes the mean squared error for translation and the quaternion distance between estimate and groundtruth in the SO(3) group.

$$loss = \sqrt{\sum(E_x^2 + E_y^2 + E_z^2) + \sum |q_e - q|} \tag{5.1}$$

where $E_{x...z}$ represents the computation of distance between estimate and groundtruth position. Quaternion distance is computed as the norm of the difference between estimate and groundtruth quaternions. In addition, we constrained the equation to take into account the fact that q and $-q$ encode the same rotation, only considering the smaller of the two possible distances in the loss function calculation.

5.2.1 Training Procedure and Hyperparameter grid-search

In order to successfully train the network and obtain a robust representation of the input data feature space, and given that there was no prior knowledge about how to tune a pose optimization network, we adopted a grid-search learning scheme to sweep multiple combinations of hyperparameters and return the one that converges to smaller loss values. This is only feasible in a short timeframe because we are working with low dimensional data (i.e. dataframes instead of high resolution imagery) but for this application, it is perfectly suited for finding an optimal solution for hyperparameter tuning.

Table 5.1: Hyperparameter grid-search setup

Hyperparameter	**range**
batch size	[4,16,32]
LSTM hidden size	[16,32,64,128]
dropout	[0, 0.3, 0.5]
loss function	'our_custom_loss'
learning rate	[0.0001, 0.0005, 0.001, 0.003, 0.005]
optimizer	['SGD' , 'RMSprop' , 'Adagrad' , 'Adam' , 'Adamax' , 'Nadam']

The grid search procedure was set up in Keras using python package Talos, on the same Nvidia GTX 1080 used previously. Note that "patience" was defined as 150 epochs, which means that there is no hard coded number of training epochs but if the loss is not decreasing for 150 epochs the model is considered trained and training is stopped. Running this grid-search procedure yielded the theoretical perfect combination of hyperparameters consisting of:

Table 5.2: Theoretical best fit for hyperparameter tuning

Hyperparameter	**Grid-Searched Tuning**
batch size	32
LSTM hidden size	32
dropout	0.5
learning rate	0.001
optimizer	'Adam'

5.2.2 Results

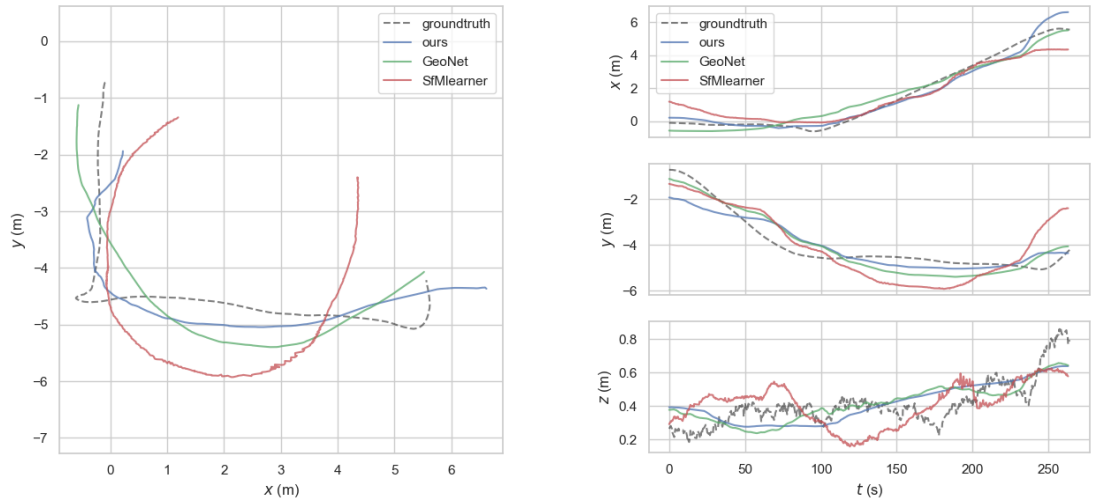


Figure 5.1: Results for the CRAS pool sequence: on the left image we can see the computed trajectory estimates against groundtruth data and on the right hand side the error decoupled by translation axis

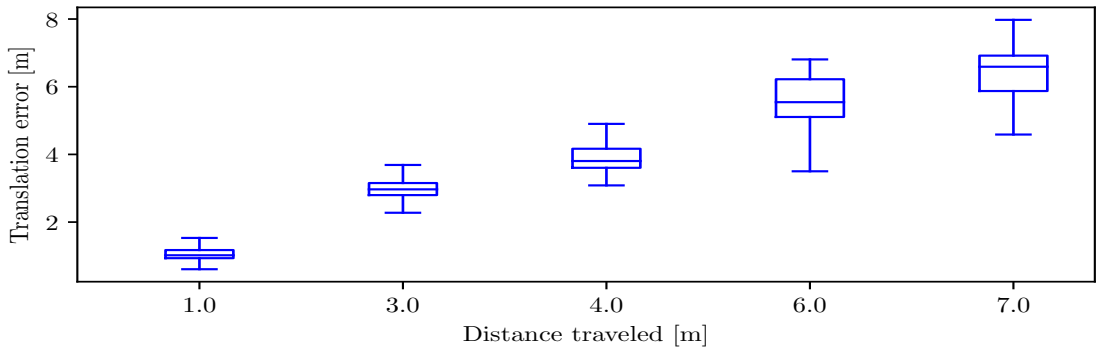


Figure 5.2: Translation Error with respect to distance traveled from our network pose estimates in our CRAS pool test sequence

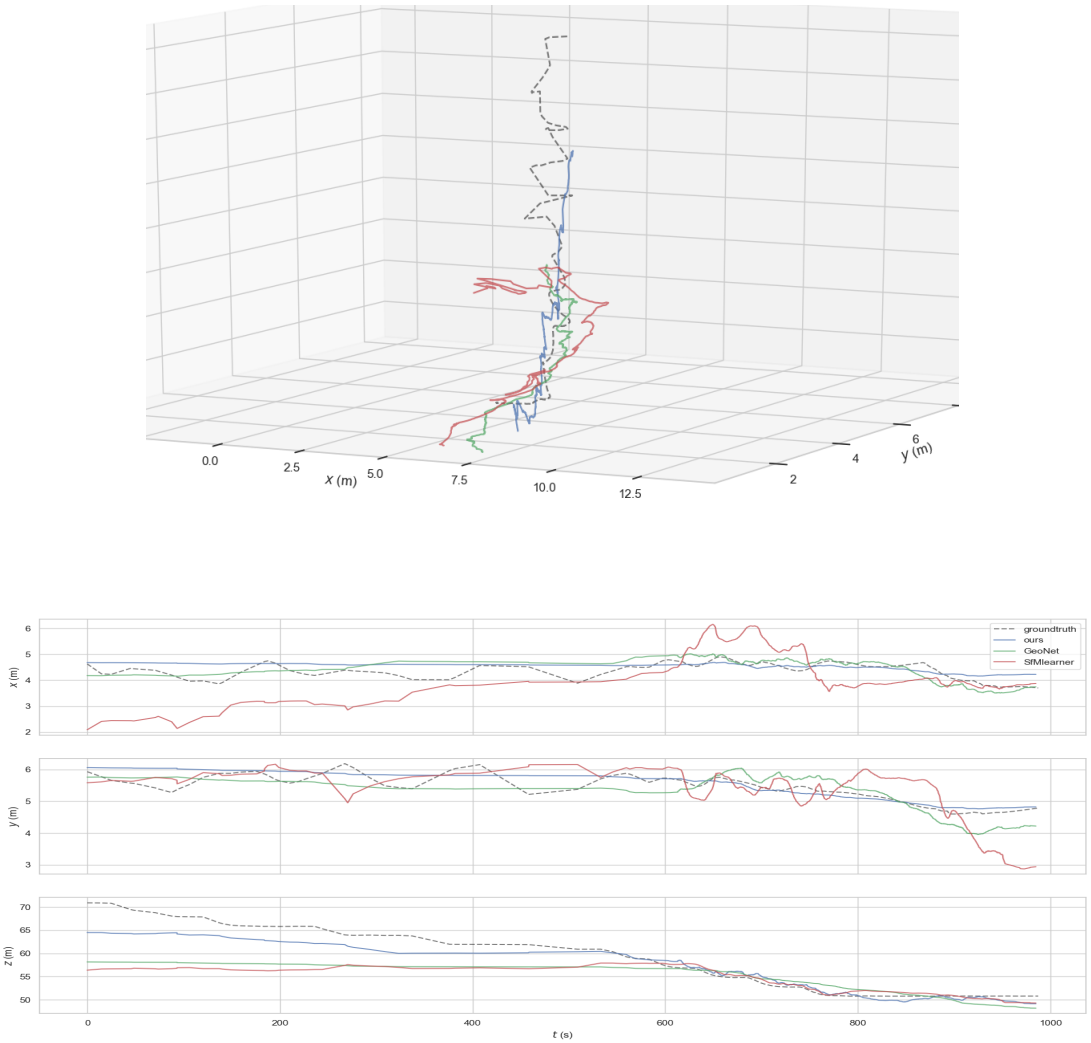


Figure 5.3: Results for Urgeiriça mine sequence: on the top image we can see the computed trajectory estimates against groundtruth data and on the bottom image the error decoupled by translation axis

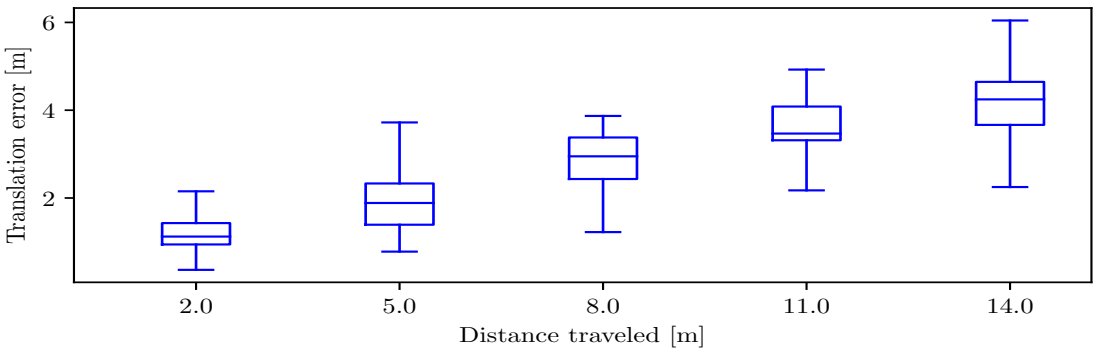


Figure 5.4: Translation Error with respect to distance traveled from our network pose estimates in our Urgeiriça mine test sequence

Table 5.3: Result compilation for absolute position error comprising both studied egomotion estimation algorithms and our visual-inertial fusion network

		Absolute Position Error (APE)					
		"raw" comparison		scale-corrected		SIM(3) Umeyama aligment	
		Avg.Error	RMSE (m)	Avg.Error	RMSE (m)	Avg.Error	RMSE (m)
CRAS POOL	SfMlearner	3.301±2.049	3.996	2.755±1.573	3.049	0.731±0.440	0.905
	GeoNet	28.739±14.613	29.912	20.846±6.687	20.087	5.345±1.112	5.475
	ours	**2.329±1.781**	**2.877**	**1.380±1.259**	**1.380**	**0.570±1.005**	**0.637**
Urgeiriça Mine	SfMlearner	52.709±1.199	52.461	20.354±3.366	19.129	0.7208±0.584	1.158
	GeoNet	55.392±2.728	56.096	22.043±1.041	22.475	0.839±0.543	1.077
	ours	**46.269±2.928**	**47.973**	**4.177±0.219**	**4.227**	**0.168±0.106**	**0.212**

As it can be observed, our Visual-Inertial Fusion Network was able to synthesize the best results for global trajectory estimation with or without any type of preprocessing step. It performs on average around **40%** better for the CRAS pool sequence while showing an average improvement of around **55%** in the Urgeiriça mine sequence. It is important to note, however, that both SfMlearner and GeoNet are unsupervised frameworks, and the divised solution leverages a supervised learning scheme.

5.3 Monocular Camera Pose Estimate Fusion Network

The second pose optimization approach dwelt on this thesis works in a very similar manner despite of its conceptual difference. Here, the proposed solution consists of fusing pose estimates from multiple camera systems so as to refine the robot global trajectory estimate. The motivation behind the development of this approach was to try to emulate stereo vision capabilities thought the use of a Recurrent Neural Network without actually modelling the stereo vision system.

This approach would be extremely interesting for the UX-1, especially because it is equipped with 5 different camera systems with non-overlapping Fields-of-View (FOV), which complicates the implementation of classical stereo vision algorithms.

It would be especially interesting if the pose optimization objective could be achieved within an unsupervised learning scheme, as that would mean we could maintain end-to-end unsupervised training and inference. This research direction was the first approach to the problem, however we did not manage, in this short timeframe, to achieve a converging model using such unsupervised learning architecture.

Despite of that, the idea of multiple camera system fusion still seemed a worthwhile concept to explore further. As such, we applied it in a Recurrent Neural Network with pose groundtruth supervision, in a similar sequence-to-sequence approach to how we did with our Visual-Inertial Fusion Network.

The goal was to understand the extent to which the integration of multiple camera pose estimates generated by deep learning architectures would influence the accuracy and performance of the predicted trajectory.

The network architecture is a Recurrent Neural Network that takes both trajectory estimates and fuses the information with groundtruth supervision. Theoretically, it would be possible to correct accumulated drift and possible inherent bias introduced by the aforementioned deep learning methods since we are estimating the same robot motion from multiple different views, therefore extracting feature information from different pool or mine walls.

The training procedure was highly influenced by the knowledge acquired in the design of the previous architecture. A more confined hyperparameter sweep was performed to tune the network for this task and otherwise the training scheme was roughly identical.

The conclusion reached was that we were not able to improve upon the standalone results from the egomotion estimation frameworks using this approach, at least with the current form of network architecture developed in this thesis frame. Nonetheless, it is worth reinforcing the merit of the concept, as if it to be successfully implemented, has the potential to effectively introduce a stereo perspective to a set of non-overlapping camera fields-of-view.

5.3.1 Results

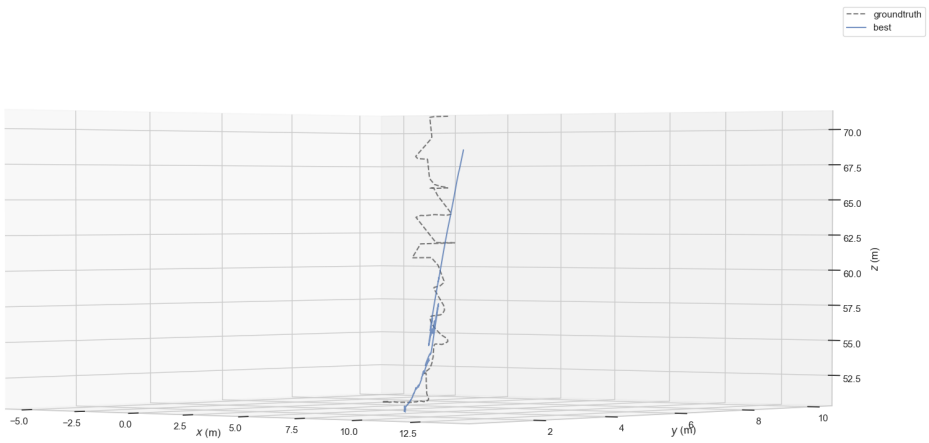


Figure 5.5: Computed trajectory estimates against groundtruth data, in the Urgeiriça mine sequence.

Table 5.4: Results of our novel multi-camera fusion network for absolute position error.

	Absolute Position Error (APE)					
	"raw" comparison		scale-corrected		SIM(3) Umeyama aligment	
	Avg.Error	RMSE (m)	Avg.Error	RMSE (m)	Avg.Error	RMSE (m)
Urgeiriça Mine	49.675±2.908	51.50	7.697±0.188	7.675	0.138±0.121	0.201

Though it can be reported that trajectory error decreased substantially, a more thorough analysis of the trajectory leads to the conclusion that the lower error with respect to translation comes to the detriment of rotation performance, with the algorithm becoming less responsive to perturbations in directions other than the most significant motion direction. Due to the nature of the data, and since error minimization is the loss minimization criterium, it can be argued that the lower error rate in the translation part of the trajectory does not represent an objectively better performance of the algorithm.

Furthermore, there is also a possibility of existence of overfitting to the training data, since the Urgeiriça mine sequence training data has a couple instances of shaft descent motions.

5.4 Summary

In this chapter we described the work done in the scope of this thesis, with respect to global pose optimization of trajectory estimates generated from the previously studied egomotion estimation frameworks.

We introduced a visual-inertial fusion network, anchored on a recurrent neural network architecture with an inertial supervision learning scheme. It was shown to improve results an average of **50%** for trajectory estimates, also producing more visually consistent trajectory estimates for both our application scenarios. This approach can later be integrated with egomotion estimation frameworks in an end-to-end fashion, leading to more accurate and reliable robot trajectory estimates.

Additionally, we described another possible solution that was dwelt on during this thesis. The implementation of a multiple camera sensor network fusion was the focus of our development in the later stages of the thesis, not ending up improving upon the previously obtained results in a meaningful way within this thesis development timeframe.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, the focus was placed on Deep Learning approaches for visual-based robot navigation, with particular interest on evaluating the potential for learning-based visual method application on complex underwater operational mission scenarios.

Firstly, a comprehensive review of state-of-the-art Deep Learning approaches for Visual Odometry applications was conducted, detailing the progress in performance and accuracy deep learning methods have managed to achieve in recent years, as well as its shortcomings. It was concluded that there was close to no information about the performance of deep learning methods in underwater context scenarios, and would therefore be particularly interesting and relevant to assess the performance of some of the most renown state-of-the-art algorithms in operational mission underwater scenarios.

The next step was to construct a comprehensive dataset encompassing different texture environments and providing different types of challenges to visual-based pose and/or motion estimation. As reported in chapter 4, this was achieved through the use of data acquired with the UX-1 robot, presenting three novel image sequences that all pose different challenges to visual-based VO estimation.

In order to access the performance of learning-based visual methods on our dataset image sequence, we focused on two different tasks: absolute relocalization and egomotion estimation. We came to the conclusion that relocalization algorithms have an overall good performance across different scenarios, but lack generalization ability when exposed to more than one different mapping during training. It is reasonable to assume that we could achieve good performance from the application of this methods in real robotic solutions, though real time testing was not performed and thus validation is still required.

As for egomotion estimation, the results were not as accurate and reliable as expected. Relative motion estimates of state-of-the-art algorithms show small errors in translation yet rotations still pose some challenges these methods are not able to overcome. Analyzing concatenated trajectories, we can easily observe that pure rotations and accumulated drifts lead to failures in pose estimation, thus making the algorithm unable to provide consistent and reliable estimates, as required by real robotic systems. In conclusion, the studied deep learning approaches for egomotion estimation do not yet meet the requirements for stable real robot implementation.

In chapter 5, we again address the issue of the aforementioned poor performance of egomotion estimation methods, presenting two possible solutions for obtaining the global pose optimization objective. The first conceived and developed solution was a Visual-Inertial Fusion Network, aimed at improving global pose estimates through an inertial supervision learning scheme. This supervised architecture proved to significantly improve results on global pose estimation, with around **50%** better error rates. Lastly, we conceived a solution where multiple camera pose estimates are fused, as a way to go around the UX-1 design constraints, where visual stereo implementations are hard to design, due to non-overlapping camera fields-of-view. However, in the timeframe of this thesis, we were unable to obtain a converging model with consistent results for this architecture. Nonetheless, it still remains as an interesting research direction for future implementations.

In this thesis, real-time implementation of deep learning algorithms was not addressed, mainly because the UX-1 does not possess any type of GPU hardware, therefore rendering any conclusion from on board implementations non-viable. It is however safe to assume that an extra real-time optimization step would need to be performed on the inference process to obtain a reasonable frame rate.

6.2 Future Work

The following future work in this thesis research scope is suggested:

- Integration of visual-inertial fusion within end-to-end deep learning for robot navigation pipelines. Further study of inertial integration without losing the unsupervised learning objective.
- Assessment and testing of visual stereo implementations on top of Deep Learning architectures. This work focused on monocular camera setups mostly due to the UX-1 design constraints, yet it would be interesting to investigate the performance of deep learning architectures for the stereo use case.
- Real-time implementation and testing of deep learning architectures for both relocalization and egomotion tasks. The low budget recommended option would be using a Nvidia Jetson Nano and TensorRT for fast inference implementation.
- Expansion of the formulation of a multiple camera pose estimate fusion scheme, that is able to regress consistent and reliable pose estimates through combining visual odometry estimates from non-overlapping camera fields-of-view

References

[1] Viable Alternative Mine Operating System - INESC TEC. URL: https://www.inesctec.pt/en/projects/vamos#intro.

[2] Autonomous Underwater Explorer for Flooded Mines - INESC TEC. URL: https://www.inesctec.pt/en/projects/unexmin.

[3] Koichiro Yamaguchi, Takeo Kato, and Yoshiki Ninomiya. Vehicle ego-motion estimation and moving object detection using a monocular camera. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 610–613. IEEE, 2006.

[4] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004. URL: https://www.engineeringvillage.com/share/document.url?mid=inspec_480457fff4e3c658M603419255120119&database=ins.

[5] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. IEEE, 2008.

[6] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[7] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Publishing Company, Incorporated, 1st edition, 2013.

[8] Hans Peter Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford, CA, USA, 1980.

[9] Clark F Olson, Larry H Matthies, Marcel Schoppers, and Mark W Maimone. Robust stereo ego-motion for long distance navigation. In *cvpr*, page 2453. IEEE, 2000.

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Yang Cheng, M. Maimone, and L. Matthies. Visual Odometry on the Mars Exploration Rovers. *2005 IEEE International Conference on Systems, Man and Cybernetics*, 1:903–910. URL: http://ieeexplore.ieee.org/document/1571261/, doi:10.1109/ICSMC.2005.1571261.

[12] Davide Scaramuzza and Friedrich Fraundorfer. Tutorial: Visual odometry. *IEEE Robotics and Automation Magazine*, 18(4):80–92, 2011. doi:10.1109/MRA.2011.943233.

[13] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry: Part II - Matching, Robustness, and Applications. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6096039, doi:10.1109/MRA.2011.943233.

[14] Peter Corke, Dennis Strelow, and Sanjiv Singh. Omnidirectional visual odometry for a planetary rover. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 4007–4012. IEEE, 2004.

[15] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[16] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[17] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[18] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[19] Hugo Miguel. INSTITUTO SUPERIOR TÉCNICO A Probabilistic Approach for Stereo Visual Egomotion.

[20] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[21] John Jin Keat Lim and others. Egomotion estimation with large field-of-view vision. 2010.

[22] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[23] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1):67–81, 1995.

[24] Berthold K P Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[25] David J Heeger. Optical flow using spatiotemporal filters. *International journal of computer vision*, 1(4):279–302, 1988.

[26] David J Fleet and Allan D Jepson. Computation of component image velocity from local phase information. *International journal of computer vision*, 5(1):77–104, 1990.

[27] Masami Ogata and Takao Sato. Motion-detection model with two stages: Spatiotemporal filtering and feature matching. *JOSA A*, 9(3):377–387, 1992.

[28] Arthur Goshtasby, Stuart H Gage, and Jon F Bartholic. A two-stage cross correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3):374–378, 1984.

[29] Theodore A Camus. Method for real time correlation of stereo images, 2003.

[30] Hugo Silva, Alexandre Bernardino, and Eduardo Silva. A voting method for stereo egomotion estimation. (June):1–16, 2017. doi:10.1177/1729881417710795.

[31] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4293–4299. Ieee, 2009.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[33] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[34] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[35] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[36] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.

[37] Vignesh Prasad and Brojeshwar Bhowmick. SfMLearner++: Learning Monocular Depth & Ego-Motion using Meaningful Geometric Constraints. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2087–2096. IEEE, 2019.

[38] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[39] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[40] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[41] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[42] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017.

[43] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using LSTMs for structured feature correlation. In *ICCV*, 2017. URL: https://github.com/NavVisResearch/NavVis-Indoor-Dataset.

[44] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[45] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO : Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. pages 2043–2050, 2017.

[46] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. 2017. URL: http://arxiv.org/abs/1709.06841, doi:10.1109/ICRA.2018.8461251.

[47] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6939–6946. IEEE, 2018.

[48] Valentin Peretroukhin, Brandon Wagstaff, Matthew Giamou, and Jonathan Kelly. Probabilistic Regression of Rotations using Quaternion Averaging and a Deep Multi-Headed Network. *arXiv preprint arXiv:1904.03182*, 2019.

[49] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. 2017. doi:10.1109/ISMAR.2016.19.

[50] Sudeep Pillai and John J Leonard. Towards visual ego-motion learning in robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5533–5540. IEEE, 2017.

[51] Mehmet Turan, Jahanzaib Shabbir, Helder Araujo, Ender Konukoglu, and Metin Sitti. A deep learning based fusion of RGB camera information and magnetic localization information for endoscopic capsule robots. *International journal of intelligent robotics and applications*, 1(4):442–450, 2017.

[52] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[53] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[54] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[55] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-D camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179. IEEE, 2013.

[56] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[57] Chris Engels, Henrik Stewénius, and David Nistér. Bundle adjustment rules. *Photogrammetric computer vision*, 2(2006), 2006.

[58] Simon J D Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, New York, NY, USA, 1st edition, 2012.

[59] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[60] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[61] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133, 1981.

[62] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. 2011.

[63] William Rowan Hamilton. *Elements of Quaternions*. Cambridge Library Collection - Mathematics. Cambridge University Press. doi:10.1017/CBO9780511707162.

[64] Robert H Wurtz. Recounting the impact of Hubel and Wiesel. *The Journal of physiology*, 587(12):2817–2823, 2009.

[65] Sepp Hochreiter and Jürgen Schmidhuber. LSTM can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[66] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.

[67] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM. \url{https://github.com/MichaelGrupp/evo}, 2017.

[68] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):376–380, 1991.

[69] Morgan Quigley, Ken Conley, Brian P Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[70] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[71] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[72] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[73] François Chollet. keras. \url{https://github.com/fchollet/keras}, 2015.

[74] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[75] Mart\'\in~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, Greg~S.~Corrado, Andy~Davis, Jeffrey~Dean, Matthieu~Devin, Sanjay~Ghemawat, Ian~Goodfellow, Andrew~Harp, Geoffrey~Irving, Michael~Isard, Yangqing Jia, Rafal~Jozefowicz, Lukasz~Kaiser, Manjunath~Kudlur, Josh~Levenberg, Dandelion~Mané, Rajat~Monga, Sherry~Moore, Derek~Murray, Chris~Olah, Mike~Schuster, Jonathon~Shlens, Benoit~Steiner, Ilya~Sutskever, Kunal~Talwar, Paul~Tucker, Vincent~Vanhoucke, Vijay~Vasudevan, Fernanda~Viégas, Oriol~Vinyals, Pete~Warden, Martin~Wattenberg, Martin~Wicke, Yuan~Yu, and Xiaoqiang~Zheng. {TensorFlow}: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL: https://www.tensorflow.org/.