Underwater Motion Estimation Based on Acoustic Images and Deep Learning

José Enrique Almanza Medina

Doctor of Philosophy

University of York

Physics, Engineering and Technology

August 2022

Abstract

This work develops techniques to estimate the motion of an underwater vehicle by processing acoustic images using deep learning (DL). For this, an underwater sonar simulator based on ray-tracing is designed and implemented. The simulator provides the ground truth data to train and validate proposed techniques. Several DL networks are implemented and compared to identify the most suitable for motion estimation using sonar images. The DL methods showed a much lower computation time and more accurate motion estimates compared to a deterministic algorithm. Further improvements of the DL methods are investigated by preprocessing the data before feeding it to the DL network. One technique converts sonar images into vectors by adding up the pixels in each row. This reduces the size of the DL networks. This technique showed significant reduction in the computation time of up to 10 times compared to techniques that use images. Another preprocessing technique divides the field of view (FoV) of a simulated sonar into four quadrants. An image is generated from each quadrant. This is combined with the vector technique by converting the images into vectors and grouping them together as the input of the DL network. The FoV division approach showed a high accuracy compared to using the whole FoV or different portions of it. Another motion estimation method presented in this work is enabled by full-duplex operation and rather than using images, it is based on DL analysis of time variation of complex-valued channel impulse responses. This technique can significantly reduce the acoustic hardware and processing complexity of the DL network and obtain a higher motion estimation accuracy, compared with techniques based on the processing of sonar images. The navigation accuracy of all the techniques is further illustrated by examples of estimation of complex trajectories using simulated and real data.

Table of contents

List of tables

List of figures

Acknowledgements

I would like to thank my supervisors, Dr. Yuriy Zakharov and Dr. Benjamin Henson, for their guidance and encouragement during my Ph.D study.

I would like to thank my thesis advisor, Prof. Paul Mitchell, for the valuable discussions and feedback on my work.

I would also like to thank the Mexican National Council of Science and Technology (CONACyT) for the financial support of my studies.

Thanks to the colleagues at the Department of Electronic Engineering for sharing great moments and experiences that helped me grow as a person. Thanks to the members of the Underwater Acoustics Research Group for the recommendations to my research and for sharing their experience.

Finally, I would like to thank Prof. Y. Petillot, School of EPS, Heriot-Watt University and Dr. N. Hurtós for providing the ship's hull data set; Dr. Luis A. Conti, University of São Paulo and Acquest Subaquatic Geology and Geophysics for providing the dam wall inspection data.

To my wife Susana, whose support is invaluable.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as references.

Publications associated with thesis work

- J. E. Almanza-Medina, B. T. Henson, and Y. V. Zakharov, "Imaging sonar simulator for assessment of image registration techniques," in *MTS/IEEE OCEANS Seattle*, 2019, pp. 1–7.
- J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, "Deep learning architectures for navigation using forward looking sonar images," *IEEE Access*, vol. 9, pp. 33 880–33 896, 2021.
- J. E. Almanza-Medina, B. Henson, and Y. Zakharov, "Motion estimation of an underwater platform using images from two sonar sensors," *TechRxiv preprint techrxiv.15059991.v1*, 2021.
- J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, "Sonar FoV segmentation for motion estimation using DL networks," *IEEE Access*, vol. 10, pp. 25 591–25 604, 2022.
- J. E. Almanza-Medina, B. Henson, L. Shen, and Y. V. Zakharov, "Motion estimation of underwater platforms using impulse responses from the seafloor," *IEEE Access*, vol. 10, pp. 127 047–127 060, 2022.

José Enrique Almanza Medina

August 2022

Chapter 1

Introduction

1.1 Overview

Almost three quarters of the Earth's surface is covered by water but only a small part of the underwater environment has been explored [10–13]. In recent years there has been a growing interest in underwater environments and considerable effort in creating and developing technologies to explore and exploit them [14].

Non-piloted underwater vehicles, such as autonomous underwater vehicles (AUVs) and remotely operated underwater vehicles (ROVs), have become an essential tool in exploration and surveying of underwater environments [15, 16]. These vehicles give significant assistance in recognition tasks [17, 18] and alleviate the dangers that humans are exposed to during exploration. Applications requiring underwater navigation include exploration of the deep ocean [19], naval surveillance [20], aquatic habitat surveys [21], dam, harbor and ship inspections, military actions [22], underwater infrastructure inspection [23], etc.

To succeed in their tasks and operate correctly, it is crucial that underwater vehicles have an accurate navigation system [11, 16, 24–26]. However, navigation underwater is still a challenge and is an active area of research [27]. Accurate detection, identification and localization of underwater objects is limited by the conditions experienced underwater. The main constraints include poor visibility, poor propagation of radio waves and constant motion

of the water body [28–30]. Under these circumstances, acoustic waves can travel through the water regardless of the water transparency and work effectively at larger ranges.

Deep Learning (DL) techniques for navigation using acoustic signals is still an area to be explored given that most of the studies related to underwater applications have been focused on object classification [31–35]. The use of DL techniques applied in image, video and audio processing [36, 37] has lead to the solution of complex problems where deterministic and other artificial intelligence techniques have been insufficient.

The aim of this work is to develop a system for motion estimation of an underwater vehicle based on processing of acoustic signals using DL which can give a high accuracy and fast processing operation to succeed in real-time applications while keeping a low hardware implementation cost.

1.2 Review of existing techniques for motion estimation

In this section, existing techniques for underwater navigation are reviewed and their advantages and limitations are described.

Underwater navigation is the process of acquiring the position of an underwater vehicle or platform and velocity in absolute or relative coordinates with respect to the environment as it moves through the body of water (Fig. 1.1) [26, 38].

The navigation problem can be addressed on a large scale (macronavigation) and on a small scale (micronavigation). Technologies for macronavigation such as the global positioning system (GPS) do not work underwater as they do in land-based applications since radiowaves are highly attenuated when passing through water bodies [39, 40].

Micronavigation techniques have been implemented for applications underwater and they can be divided into three types [41]: (i) techniques based on beacons, (ii) inertial and dead reckoning systems and, (iii) techniques based on environment referencing.

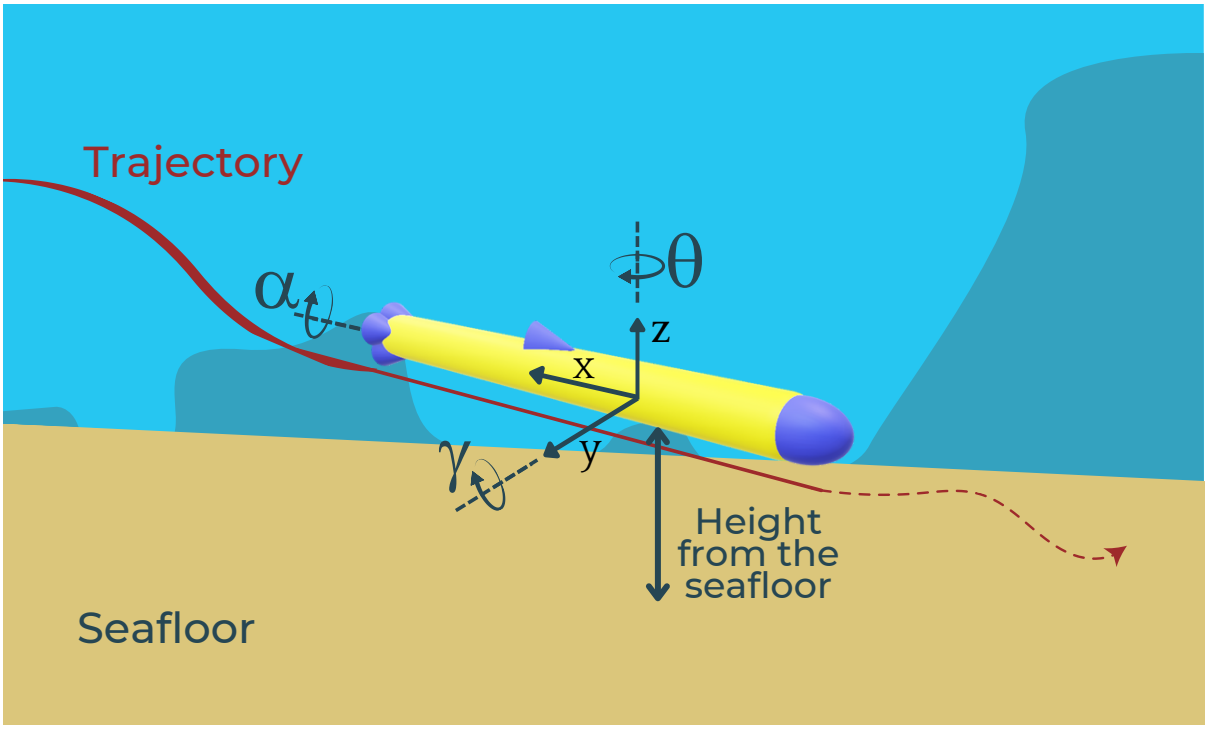
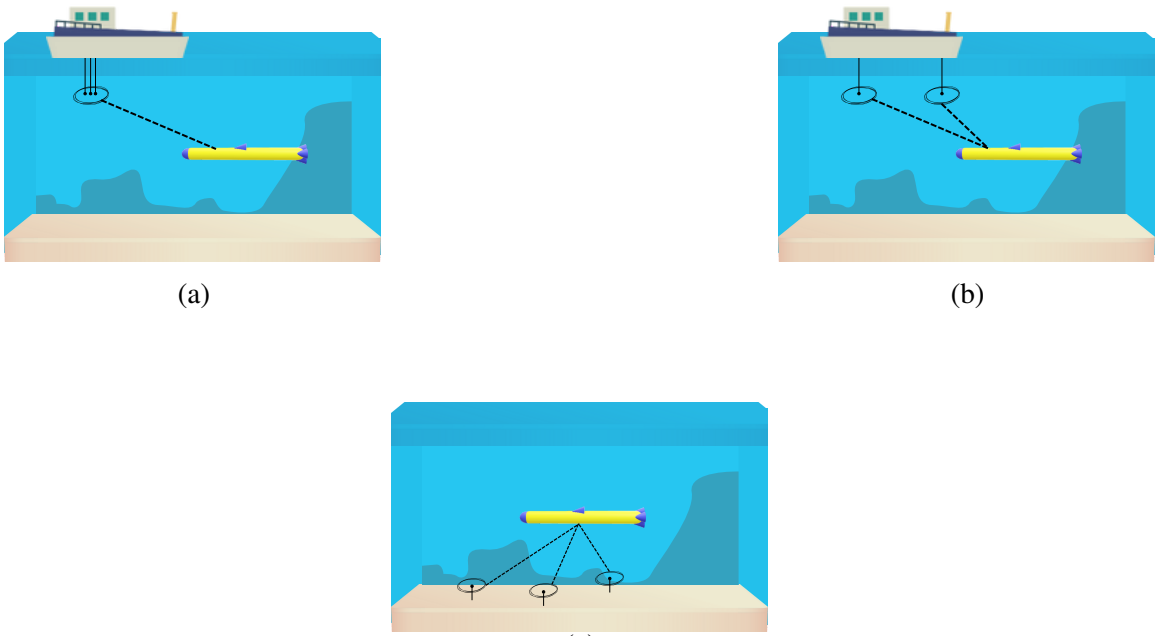


Fig. 1.1 Underwater vehicle follows a trajectory. The navigation system calculates its position and orientation.



(a) (b)

(c)

Fig. 1.2 (a) Ultrashort Baseline. (b) Short Baseline. (c) Long Baseline.

1.2.1 Techniques based on beacons

This type of navigation requires the use of nodes at fixed positions that send signals (acoustic beacons) used by the underwater vehicle to calculate its position [42]. The long baseline (LBL), short baseline (SBL) and ultra short baseline (USBL) systems fall in this type of navigation [43, 44].

A USBL system uses a transceiver with three or more transducers, separated by a minimum of 10 cm and a transpoder (Fig. 1.2a). The transceiver is placed on the bottom of a ship on the sea surface and the transponder is placed on the AUV. The AUV position is calculated by measuring the relative directions and ranges (distances) from the ship to the AUV. The relative direction is obtained from the phase difference in the received signals [44, 45]. SBL systems (Fig. 1.2b) operate in the same way that USBL systems do, with the difference that the minimum separation of the transducers is longer than that in USBL. In SBL, the relative directions are obtained by measuring the time of arrival of signals from different transducers [44]. In a LBL system (Fig. 1.2c), an AUV calculates its position by triangulating the signals from three beacons fixed on the seafloor [46].

The major constraints of these techniques are that calibration and deployment can be problematic [47, 48], that navigation is only possible in a limited geographical area covered by the beacons [49–51] and it also makes the system complexity and cost high [52]. Also, USBL systems require a high energy consumption [53].

Another technique that uses beacons is presented in the work [54], which considers deploying buoys equipped with GPS. The buoys transmit their positions to users or devices underwater to calculate the user position through triangulation. Similarly, the work [55] proposes a method that uses a constellation of underwater acoustic devices with known positions to estimate the position of underwater vehicles.

1.2.2 Inertial and dead reckoning systems

An inertial navigation system (INS) uses accelerometers and gyroscopes to estimate the vehicle position, velocity and attitude [56].However, an INS suffers from an unavoidable

gyroscope drift and accelerometer bias [57, 58], which can be compensated by the fusion with other sensors such as the Doppler velocity log (DVL) [59, 60]. Also, an INS needs the vehicle to return to the surface for resetting its position [55, 61]. A DVL uses a bottom-referenced acoustic navigation method that consists in transmitting narrow beams from three or four sensors in the direction of the seafloor at different angles and the vehicle velocity is calculated from the measured Doppler effect. To produce the narrow beams, high frequencies are required and therefore the acoustic signals are highly attenuated, limiting its usage to shallow waters [62]. The DVL is highly sensitive to the external environment conditions [63], it is expensive and not suitable for mounting on many vehicles due to its size [64]. A correlation velocity log (CVL) is less constrained in this respect. A CVL has a downward looking single wide-beam projector and an array of hydrophones to receive the backscattered signals from the seafloor. This allows the use of lower frequencies with less signal attenuation [65].

Dead reckoning is a simple method in which an underwater vehicle can calculate its position based on its velocity and time. It requires a water speed sensor and a compass to obtain the direction and vehicle's velocity, respectively. The main constraint of dead reckoning is the water current. It affects the speed sensor by increasing or reducing the current velocity measurement (see Fig. 1.3) [1].

To overcome the limitations of dead reckoning, some authors develop techniques to aid and improve the navigation with dead reckoning. For instance, the work [66] proposes a combination of sensors (INS with a pressure sensor) and the use of a recurrent neural network (RNN). The RNN network helps to correct the navigation measurements from the sensors readings. In [67], compensation to dead reckoning is applied using a long short-term memory (LSTM) neural network that estimates the sideways and forward motion of the AUV using generalized forces from different sensors and previous velocity estimates. The work [68] presents an algorithm for dead reckoning navigation which uses an extended Kalman filter to estimate the attitude, orientation, and gyroscope bias from a micro-electromechanical system.

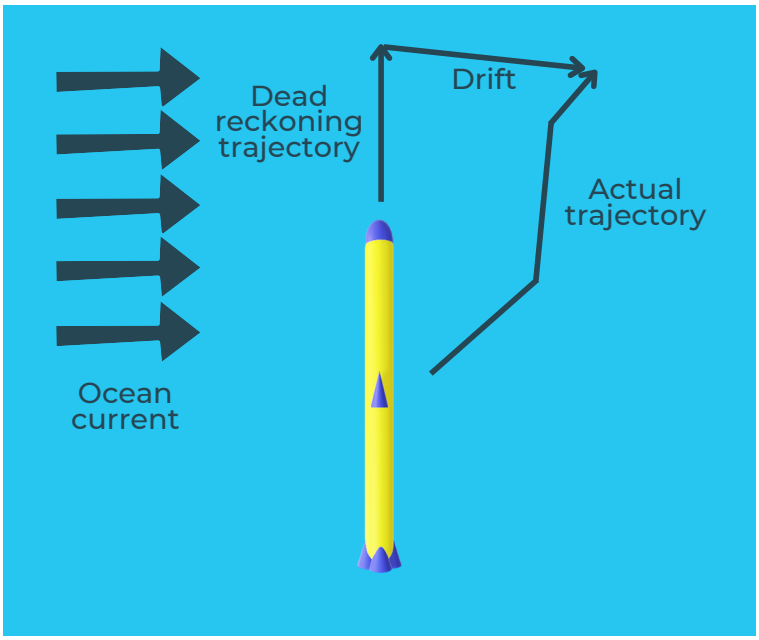


Fig. 1.3 Dead reckoning drift effect (Adapted from [1]).

1.2.3 Techniques based on environment referencing

These techniques use sensors for extracting information from the vehicles surrounding environment to estimate the position and attitude and can be divided into visual and acoustic. Motion estimation based on optical images is a well known approach in terrestrial [69, 70] and aerial [71, 72] applications. Some effort has been done underwater. For example, visual odometry [64, 73] uses images from optical cameras to estimate the underwater vehicle motion and can provide simultaneous localization and mapping [74–77] and recently applying a DL network to a sequence of images from a camera [76] has been explored to estimate the trajectory of an underwater vehicle. Optical methods are often used as a complement for other sensors, but visual based navigation methods alone are inaccurate when the visibility is poor, even in perfect conditions they only work up to around 20 m underwater [30]. Acoustic imaging techniques are not affected by these conditions and they can operate further than 100 m [30] and techniques for motion estimation using sonar images have been developed. For instance, pairs of consecutive sonar images are used to estimate the motion of an underwater vehicle by using a deterministic image registration

method [29, 78, 79]. Image registration is the process of aligning two or more images of the same scene that differ in time and pose (viewpoint and/or orientation) due to the displacement of the acquiring sensors or changes in the scenario over time. This could be less than a second or a few days [80, 81]. The method in [78] presents an image registration algorithm for motion estimation with four degrees of freedom (DoF) that uses a probabilistic model to calculate the probability that a point (pixel) in the image was generated with sonar reflections from a certain range, bearing and elevation arc in the environment. This probability is used by the algorithm for estimating the motion between two frames. The image registration method in [29] uses Forward-Looking Sonar (FLS) images to estimate the motion with three DoF. The image is processed in the frequency domain using a phase correlation technique and motion estimates are obtained from the maximum of the phase in the generated correlation matrices. The method in [79] is a deterministic algorithm for attitude and trajectory estimation with three DoF. It works with sequences of sonar images and it is capable of estimating pixel displacements between two sonar images with a sub-pixel accuracy. The pixel displacements are used to estimate the attitude and trajectory of the imaging sonar sensor. In [82], a method is presented for navigation, which consists of reducing the images from a three-dimensional (3D) multi-beam sonar to a depth map that is compared with a pre-registered reference map of the bottom.

1.2.4 Sensor combination

Underwater navigation is often performed by the fusion of multiple methods or sensors. This combination can alleviate the problems associated with the original methods but at the cost of increasing the whole system complexity, making it expensive and harder to deploy.

Some systems attempt to reduce the complexity and cost, for example [77] presents the use of two low-cost inertial sensors and a sonar (altimeter). In [83], FLS images and INS are used with an adaptive Kalman filter to replace a DVL. In a similar way, [84] uses an INS and replaces the DVL by extracting the dynamic model of the vehicle using Newton-Euler equations. In [85, 86], INS are used as an aid to other techniques such as Displaced Phase Center Antenna (DPCA) for Synthetic Aperture Sonar (SAS) micronavigation.

The work [87] proposes mixing the information from a sonar and INS. The work in [59] presents a combination of MEMS with INS and DVL. In [88], INS, DVL and USBL are combined. Furthermore, in [89], even more methods are combined, by processing data from INS, DVL, LBL, and pressure and attitude sensors. In [90], the authors propose a method that combines visual odometry with INS measurements. The work [11] proposes combining the information from aerial images with the images obtained by an FLS.

The literature review presented above explains how several underwater navigation systems have been implemented. The combination of multiple traditional techniques has aided to reduce problems associated with each individual technique. However, the system complexity and cost are still a challenge to be addressed in this thesis. Therefore, this work proposes novel underwater motion estimation techniques based on acoustic signals using DL networks and their corresponding navigation techniques that can be suitable for real-time applications while keeping a high estimation accuracy.

1.3 Contributions

The original contributions of this work are summarized as follows:

1. A simulator capable of generating realistic sonar images from 3D scenarios [5]. The simulator allows customisation of sonar parameters such as number of beams, maximum measured range and the field of view (FoV). The simulated sonar images can be noiseless or with noise. The noise model can also be adjusted.

2. A collection of data sets of simulated acoustic (sonar) images with their respective labels that represent the underwater sonar platform motion. The data sets can be used for training and validation of motion estimation techniques based on the analysis of acoustic images, e.g., techniques based on image registration.

3. Several DL networks are implemented and compared to identify the most suitable for motion estimation using sonar images [6, 7]. The DL networks are validated using noiseless and noisy images. The DL methods are also compared with a deterministic

method, showing a much lower computation time and more accurate motion estimates than the deterministic algorithm.

4. A technique that consists in converting sonar images into vectors by adding up the pixels in each row [8]. This reduces the size of the DL networks that are used to estimate the motion. This technique showed a significant reduction in the computation time compared to techniques that use images.

5. The division of the FoV of a simulated sonar into four quadrants [8]. An image is generated from each quadrant. This is combined with the vector technique by converting the images into vectors and grouping them together as the input of the DL network. The FoV division approach showed a high accuracy compared to using the whole FoV or different portions of it.

6. A motion estimation method, enabled by full-duplex operation, based on DL analysis of time variation of complex-valued channel impulse responses [9]. The full-duplex operation allows continuous transmission and reception of signals by two or four receivers.

1.4 Thesis structure

The remainder of this thesis is structured as follows:

- Chapter 2 describes the design of the imaging sonar simulator. The chapter also presents the evaluation of an image registration method for attitude-trajectory estimation using data sets generated by the simulator.

- Chapter 3 presents the analysis of DL architectures for motion estimation in three DoF using sonar images from one sonar sensor with the transmitter and the receiver at the same place and a configuration that uses images from two sensors separated from the transmitter. The chapter also presents the process to generate large volume data sets to train the DL networks and the results of trajectory estimation and mosaic building using simulated and real sonar images.

- In Chapter 4, modifications to the simulator from Chapter 2 are presented to include the possibility for segmentation of the FoV. Then a novel motion estimation method that uses different portions of the sonar FoV and a technique that compresses the sonar images into vectors are described. Results and discussion for implementing the vector technique and the FoV segmentation are presented. Trajectory estimation with simulated and real data using the proposed techniques is also presented in this chapter.

- In Chapter 5, modifications of the simulator from Chapter 2 to generate channel impulse responses are presented. Experiments are presented for motion estimation with data sets of impulse responses as inputs of a DL network. A comparison with methods that use sonar images instead of impulse responses is also presented.

- Chapter 6 presents the thesis conclusions and directions of further research work.

Chapter 2

Forward-looking sonar simulator

In this chapter, a description is presented about how to build a forward-looking sonar simulator. The aim of the simulator is to generate large volumes of ground truth data to test algorithms such as: novel image registration techniques for trajectory estimation, three-dimensional object reconstruction or to be used to produce training data for machine learning algorithms. The simulator is capable of generating realistic data sets of images and providing ground truth data with the exact position and attitude of the sonar related to objects in a test scenario which can be customized for different tasks. Section 2.1 gives an introduction into sonar simulation techniques. Section 2.2 describes the design of the imaging sonar simulator. Section 2.3 presents results and evaluation of an attitude-trajectory estimation method using data sets generated by the simulator. Finally, conclusions are given in Section 2.4.

The work in this chapter is presented in the paper: J. E. Almanza-Medina, B. T. Henson, and Y. V. Zakharov, "Imaging sonar simulator for assessment of image registration techniques," in *MTS/IEEE OCEANS Seattle*, 2019, pp. 1–7.

2.1 Introduction

As mentioned in Chapter 1, the utilization of acoustic imaging techniques presents advantages in scenarios where optical cameras have poor performance due to scarce illumination or turbidity of water [91]. Two-dimensional (2D) FLSs can generate high-resolution images.

Regardless the technical specifications defined by each manufacturer, the image formation process is mostly identical [29]. This type of image is created when an acoustic pulse is emitted from the sonar and reflected by objects in the scene. The reflections are collected by a sensor in the sonar from different aperture (azimuth) (ϵ) and elevation (ϕ) directions, revealing the distance, position and acoustic reflectivity of the objects. The information is compressed into a 2D image by losing the elevation dimension in the process (Fig. 2.1). Using polar coordinates, the intensity of a pixel on a sonar image can be represented by [92]

$$I_{pixel}(r,\epsilon) \approx \int_{\phi_1}^{\phi_2} \beta(\phi) G_s(r,\epsilon,\phi) J_s(r,\epsilon,\phi) d\phi, \tag{2.1}$$

where a single pixel is formed by the contribution of all the intensities from points in 3D space (r,ϵ,ϕ) over the elevation $[\phi_1,\phi_2]$. $\beta(\phi)$ is a function related to the beam pattern, $G_s(r,\epsilon,\phi)$ is a measure related to objects reflectivity and $J_s(r,\epsilon,\phi) = \frac{\vec{v}\cdot\vec{n}_{r\epsilon\phi}}{\|\vec{v}\|\left\|\vec{n}_{r\epsilon\phi}\right\|}$, is the cosine of the angle between the direction of the beam $\vec{v}$ and the surface normal $\vec{n}$ at point (r,ϵ,ϕ). Therefore, when an image is generated there is a transformation from a spherical coordinate system (r,ϵ and ϕ) to a two-coordinate system (polar) composed by r and ϵ only.

The development of new models and techniques based on sonar imaging often requires large collections of data taken from a controlled test scenario and accurate knowledge of position and attitude of objects and sensors in the scene. For that reason underwater simulators have been developed where all these features can be easily manipulated and tuned.

Simulating aquatic environments can be a challenging task. The process of recreating realistic data sets becomes complex when including such factors as object motion.

In [93], a novel open source simulator is presented. It uses the OpenSceneGraph programming interface and specialist libraries for realistic underwater rendering. This simulator uses Robot Operating System (ROS) for interaction between the user, vehicles and sensors. The authors created different simulation sensors, but none of them is for an imaging sonar.

In order to create test samples for a 3D reconstruction algorithm, an imaging sonar sensor was built in [94] using the underwater simulator from [93]. The simulator uses a ray-tracing technique which obtains the angle of incidence between the sonar beam and the normal to the object (Lambertian model [95, 96]). It considers a constant reflection coefficient for all

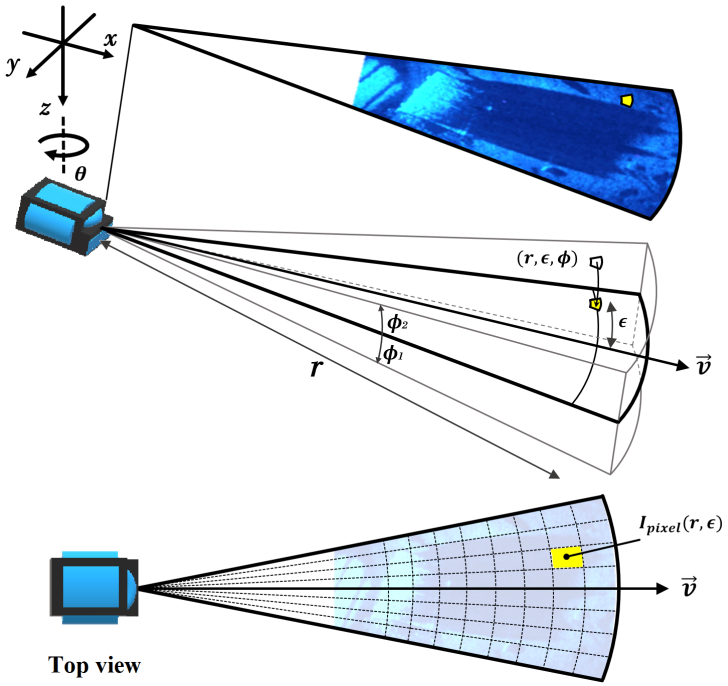


Fig. 2.1 Image formation geometry.

the objects, i.e., the intensity of the reflected beam is only affected by the angle of incidence and not by the reflectivity of the material. The beam pattern is shaped as a sinc function and noise is added according to a noise model for the BlueView P900 imaging sonar [97]. Although this simulator is provided as open source software, one of difficulties in its use is the background knowledge required to operate it, specifically, as it requires advanced skills in C++, Python and XML languages besides some experience with ROS. Furthermore, it only works on Linux systems.

Ray-tracing is a technique that creates 2D images from a 3D world. Basically it consists in tracing the trajectory of a ray from a source to a different point while it is reflected by objects in the simulated scenario [98]. Despite having a high computational cost, ray-tracing is a common initial approach for sonar imaging simulators to produce high-quality images [91, 99]. The work [100] presents a ray-tracing simulator for the design and evaluation of sonar features such as beam shape and beam location. The main applications of the simulator are the evaluation of obstacle detection and tracking methods. The simulator

is implemented in Unreal Engine to display the simulated environment and with ROS to manipulate the vehicles.

The *Rock-Gazebo framework* developed in [101] is used by [102] to build a simulator for a Mechanical Scanning Imaging Sonar (MSIS) and a FLS. The simulated sensor emits pulses into the scene and stores three parameters in memory using the 8-bit RGB channels: (i) the distance to the object; (ii) the intensity of the reflected signal, which is proportional to the incidence angle of the beam and normal to the surface; and (iii) the angle of arrival. Speckle noise (modelled by a Gaussian distribution) is applied to the final image. However, noise is added to every pixel with no distinction of the origin of the pixel, e.g., if it was generated by an acoustic shadow or an object, they are treated equally.

The work [103] proposes an approach for the development of an imaging sonar simulator that is focused on adding three types of distortion to the images. It considers the speed of sonar motion, the navigation course deviation and the optical noise on the pixels of the image. The imaging procedure uses the ray-tracing approach and no environment conditions are considered such as different values of reflectivity on the surface of the objects.

Another image simulation approach consists in a less conventional combination of ray-tracing with a frequency domain algorithm [91]. Similar to other methods, three parameters are collected to construct the image: (i) Euclidean distance from the source to points in the scene; (ii) reflected intensity; and (iii) a number associated with each of the polygonal surfaces that make the objects. The number is used to avoid overlapping (summation) of rays that return from the same area when producing the image. This reduces the computational cost. However, if an object presents a large surface or a low number of polygons, most of its surface will be ignored. There is a risk of missing information and producing incomplete images. Another feature to remark is the angle ray separation which can be specified from one of two configurations: (i) the same angular separation or (ii) the same separation on the seabed. In a similar way, another scanning strategy is described in [104]. Rays in elevation angle are non-uniformly distributed to scan only the points on the surface of the objects that offer more information of the shape. However, this ray separation approach may require greater computation when analyzing the object shape to chose the right scanning pattern.

The simulator in [105] uses two methods for sonar imaging: (i) a ray-tracing method and (ii) a method that uses the rays to create *tubes* for modelling the propagation of acoustic waves. When a tube is emitted from a transmitter, it produces a footprint on the surface of the objects. The intersection of two or more footprints from different tubes represents one path of the ray travelling in the space, allowing the consideration of multipath in simulations. This simulator also defines two concepts for textures in the objects that, when scanned, help to produce more realistic images: macro-textures, which are small deformations in the surface of the objects, and micro-textures, which are represented by variations in the colour or image attached to the surface of the object.

Summarizing the features of previous works and considering that the aim of the new desired simulator is to supply large volumes of samples to feed techniques related to the image registration, the following characteristics need to be considered:

- **Reflectivity (G_s):** It should consider material properties on the surface of the objects in the scene. Therefore macro and micro-textures similar to that in [105] should be used.
- **Angle of incidence ($\cos^{-1} J_s$):** Total reflectivity should consider the angle between rays and surface of objects.
- **Noise:** Model of noise that is present on FLS images. It should be able to be adapted or customized to the simulated sonar.
- **Computational and time consuming cost:** Since it is expected to generate large volumes of data.
- **Adaptability:** It can be easily modified to simulate different types of FLS.
- **Mobility:** Sonar motion paths must be set and controlled easily in order to scan the scenes from multiple points of view.
- **Multi-platform:** The simulator can be used in different operating systems.

The sonar simulator proposed considers the aspects mentioned above in order to produce realistic sonar images from underwater environments. The generated data set can be used for

testing image processing techniques while having a low computational cost. Equal separation between rays is used to reduce the processing time when scanning the scene rather than a non-equally spaced scheme. The present work focuses on building an imaging sonar simulator using the increasingly popular and low-cost platform Unity [106]. Unity is a well-tested development platform capable of creating 3D scenarios, providing a large programming toolset, an intuitive workspace and realistic manipulation effects controlled by its own physics engine [107–109]. The physics engine is a software program that enables simulating motions and reaction of objects under the laws of physics, in a similar way to the real world. Unity initially focuses on the creation of video games. However, it has also been used in education, medical, animation, construction design and industrial applications [110]. Project testing and editing can be done *on-the-fly*, i.e., while a simulation is running. Programming code is written in the C# language. It is possible to operate directly with the objects in the scenes allowing highly efficient management of the environment. Moreover, the Unity platform can be run on Windows, Mac and Linux operating systems.

2.2 Sonar simulator design

The sonar simulator is divided into three parts. The first part emulates the underwater scenario, defining the sonar and its motion and acquiring the data required to produce images. The second part processes the information collected in the first part to generate a set of images. The third part adds noise to the images. The whole process is summarized in Fig. 2.2.

2.2.1 Generation of underwater scenarios and data acquisition

The first part of the simulator is implemented on the Unity platform. The simulator is based on a ray-tracing technique where a series of rays are emitted from the sonar at certain angles that then collide with the objects in the scene. When a ray hits an object, it stores in memory three features needed to produce the image: (i) Euclidean distance from the sonar to the collision point, i.e., the length of the ray when it hits an object; (ii) the pixel colour information (in the RGB colour system) of the point being hit. The colour represents

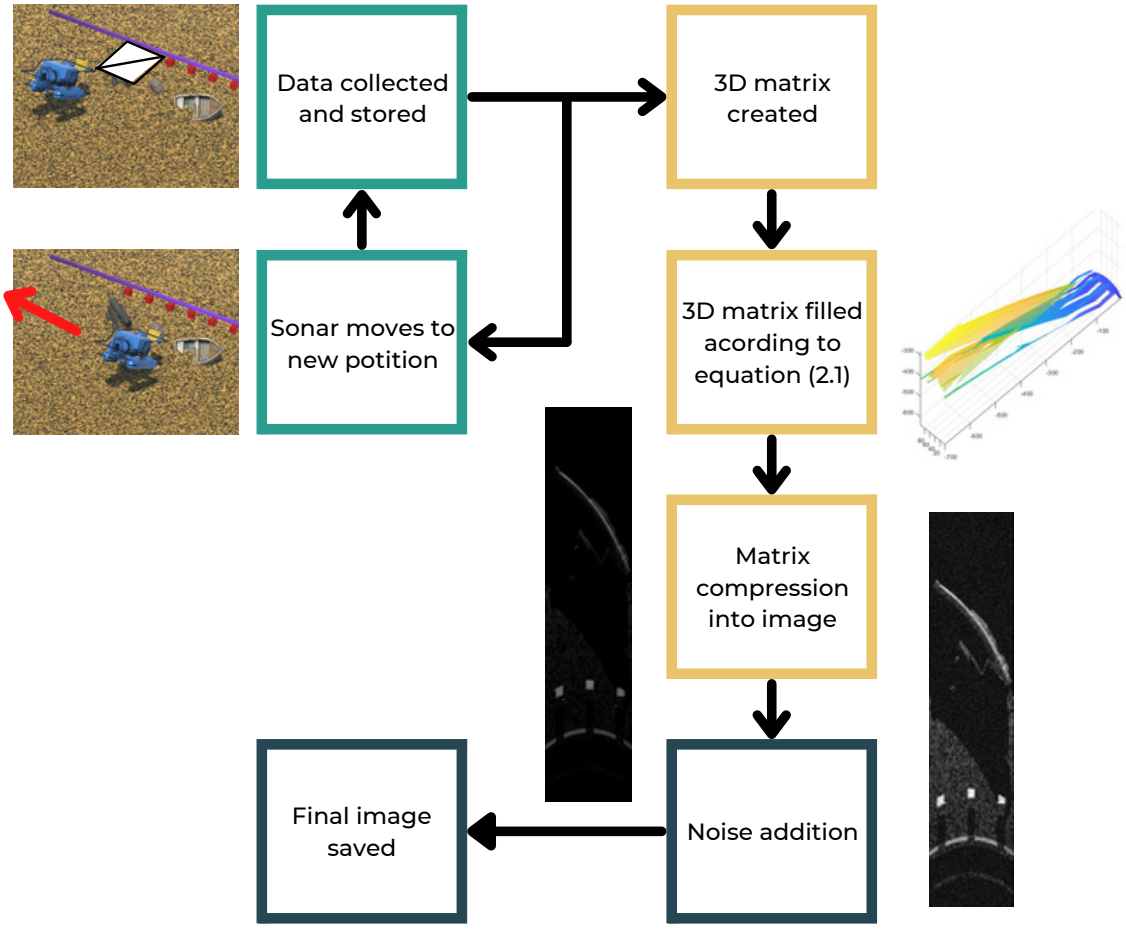


Fig. 2.2 Image formation process.

(a) (b)

Fig. 2.3 (a) Multiple rays forming a beam. Only seven rays are shown here but a larger number is used to generate images, e.g., 700. (b) Multiple beams forming the FoV of the sonar. For demonstration purposes only four beams are displayed, the exact number N_B depends on the sonar being simulated. λ represents the azimuth angle or aperture, ψ the elevation angle and R_{max} is the maximum measurable range from the sonar.

a measure of the acoustic reflectivity of the material; and (iii) the angle of incidence between the ray and the vector normal to the surface of the object following the Lambertian model.

Multiple rays are equally spaced in elevation angle over a beam (Fig. 2.3a). In order to obtain more realistic imagery within a low simulation time, a trade-off between resolution and computation/storage cost can be established.

The sonar field of view (FoV) is formed by a set of beams uniformly separated in the azimuth aperture (Fig. 2.3b). The number of beams N_B and the aperture angle λ are set according to specifications of the sonar that is emulated. The elevation angle ψ, the number of rays per beam N_R, the maximum measurable range R_{max}, position and orientation of the sensor, in six DoF, are defined in a Unity dashboard created to control the sonar. The total number of rays to be generated is $N = N_B \times N_R$.

The motion-scanning algorithm of the sonar defines a trajectory by specifying start and end points in the 3D scenario ($A_i = [a_{xi}, a_{yi}, a_{zi}]$ and $A_o = [a_{xo}, a_{yo}, a_{zo}]$, respectively) and the number of image frames to be acquired T. T points equally spaced on the trajectory are calculated and an image is obtained from each of these points. The sonar can rotate while it moves by specifying initial orientation angles $A_{rot_i} = [\alpha_i, \gamma_i, \theta_i]$ and final orientations angles $A_{rot_o} = [\alpha_o, \gamma_o, \theta_o]$, where α, γ and θ represent the rotation around x, y, and z, respectively.

For each position, the sonar stores the data that was collected by the rays in three matrices of size $N_B \times N_R$. The first matrix **R** contains the Euclidean distance from the sonar to each of the points hit by the rays. The second matrix **C** stores a value between 0 and 1. For this simulator, only the red component from RGB baseline is saved, this colour information is

used as a measure of reflectivity on the surface of the object, the redder a pixel is, the more reflective it is, and if it is darker, it absorbs more energy and its reflectivity is low. The third matrix **L** keeps the cosine of the angle of incidence between the ray and a vector normal to the surface of the object. These matrices represent the functions in (2.1), where $G_s(r, \epsilon, \phi)$ and $J_s(r, \epsilon, \phi)$ correspond to **C** and **L**, respectively, and for simplicity, the beam pattern $\beta(\phi)$ is set to one. **R** is used to map the information of the objects to the correct position in the image. The information about the position and orientation of the sonar at each point of the trajectory is also stored to be used as the ground truth.

2.2.2 Generation of acoustic images

For each position in the trajectory, an image is generated from the data acquired in the previous step. This process is described for a single frame, but it is repeated for each of T desired images. A 3D matrix **U** of size $N_R \times N_P \times N_B$ is created (elevation, range and aperture, respectively). N_P is defined by the emulated sonar and represents the number of different possible range values, i.e., the number of pixel rows in the final image. Initially, the matrix is filled with zero values and only some of its elements are replaced by an intensity value at specific coordinates that are defined by

$$U\left(\left\lceil \frac{R(i,j)N_P}{R_{max}} \right\rceil, i, j\right) = C(i,j)L(i,j), \tag{2.2}$$

where $R(i,j)$, $C(i,j)$ and $L(i,j)$ are the (i,j)th entry of the matrices **R**, **C** and **L**, respectively, i and j are indexes that represent coordinates in the image matrix and $\lceil a \rceil$ is the ceiling operation that returns the nearest integer greater than or equal to a number a.

The matrix **U** is compressed into a square matrix to create the final image $\tilde{U}$ of size $N_P \times N_B$. Compression is performed by adding all the values in each single column:

$$\tilde{U}(k,m) = \sum_{n=1}^{N_R} U(k,m,n). \tag{2.3}$$

Therefore the elevation axis of the 3D matrix is lost and only aperture and range axes remain. If resulting values from the summation exceed a threshold, the value is truncated to that

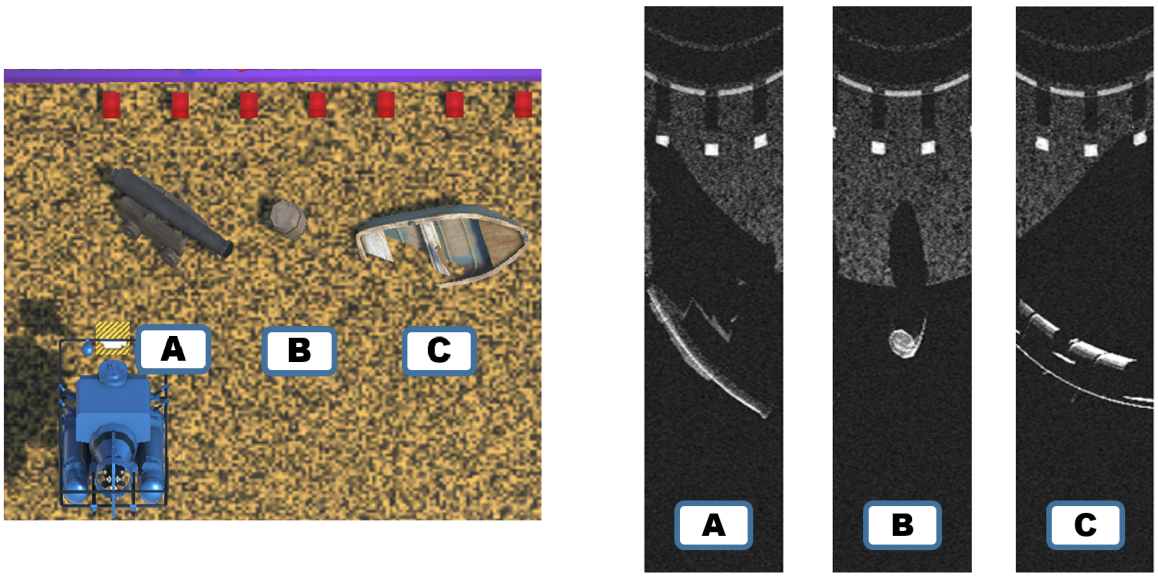


Fig. 2.4 3D testing scenario and resulting images acquired from positions A, B and C in front of a cannon, barrel and boat respectively, imitating DIDSON 300 sonar features [2].

threshold, e.g., 255, since pixel values in this type of images have a scale from 0 to 255 (from dark to bright, respectively).

2.2.3 Adding noise to images

After the image has been produced, its pixels are distorted by adding noise according to a model using the following criteria. If the pixel lacks the information about objects in the scene, i.e., it is of a zero value, then it is considered to be produced by an acoustic shadow or the water column and being affected only by electrical noise inherent to the sonar; thus the pixel value is modified according to a Gaussian distribution model [111–113]. Otherwise, if the pixel created using the information from object surfaces, the Rayleigh distribution is applied, since it corresponds to a surface represented by many small scatterers [112–115]. The parameters of the Gaussian and Rayleigh distributions are chosen according to measures obtained from real imaging sonar devices. The result of adding noise is the final image $\hat{U}$. Examples of simulated images obtained can be seen in Fig. 2.4.

2.3 Image registration using simulated images

This section presents results of applying an image registration method on multiple data sets of simulated images. Image registration is a process that compares a reference image with one or more images of a scene from different points of view at different moments in order to obtain information on the change in the imaging conditions [81]. The method for trajectory estimation and mosaicing developed in [79] is implemented. The method is capable of estimating pixel displacements between two frames with subpixel accuracy. The displacement information is used to estimate the attitude and trajectory of the imaging sonar sensor. In this estimation, three parameters (W = 3) of the sensor motion are calculated and they are represented by $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]$: translation in the x- and y-axes and rotation around the z-axis (θ), respectively, according to directions displayed in the coordinate system of Fig. 2.2. Finally, a mosaic is built using the image data set and the attitude-trajectory estimates are obtained. This method was applied on the data sets for validating its performance using the ground truth provided by the simulator.

The data sets have been constructed using three different scanning trajectories over a scene defined as follows.

1. **Trajectory 1**: Alternation between translation in the x and y directions. 250 frames along 5 m in the y direction, 100 frames along 1.2 m in the x direction, 250 frames along 5 m in the -y direction, 100 frames along 1.1 m in the x direction and 250 frames along 5 m in the y direction, 950 frames in total.

2. **Trajectory 2**: Translation in the xy direction. 300 frames along 2.3 m and 5 m in the x and y directions respectively.

3. **Trajectory 3**: Rotation over z-axis. 300 frames over $120°$ around the z-axis on the same position.

For the three trajectories, the simulated sonar has a constant rotation of $35°$ around the x-axis (α) and no rotation around the y-axis (γ).

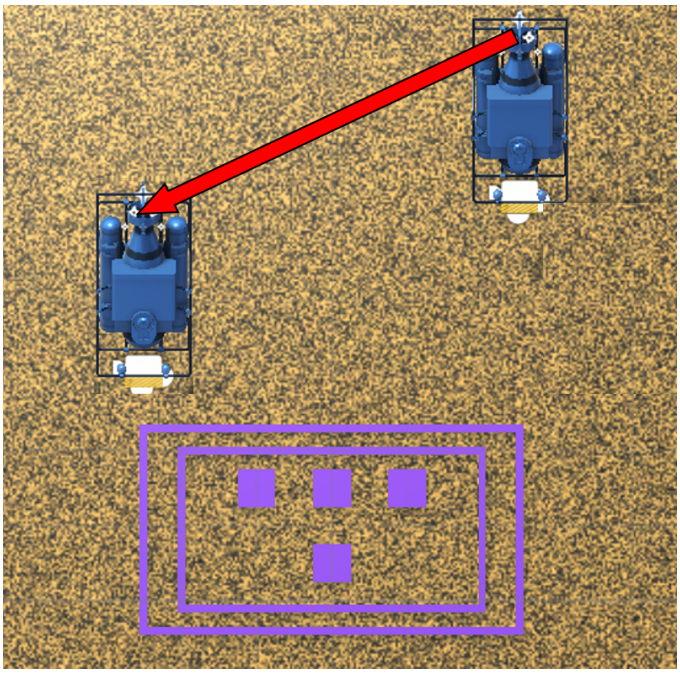
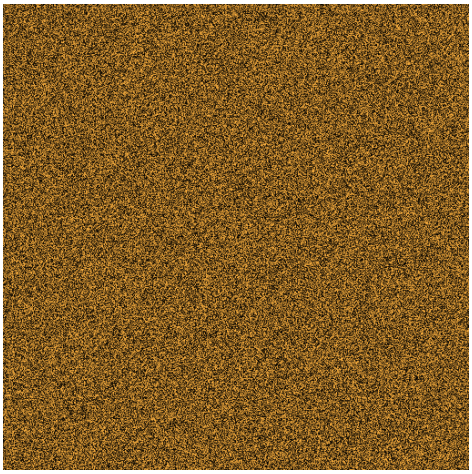


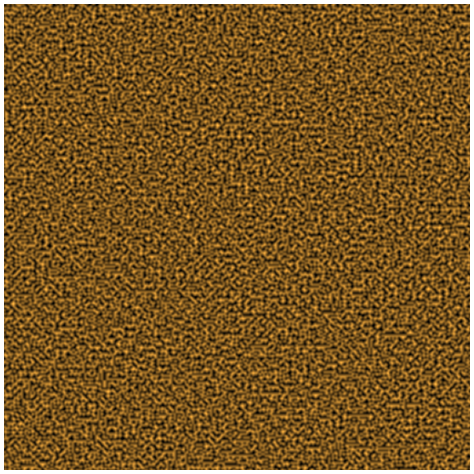
Fig. 2.5 Testing scenario seen from above. Gaussian noise as seabed texture. Red arrow represents the trajectory of the sonar moving 5 m in y-axis and 2.3 m in x-axis with no rotation.

The testing scenario can be seen in Fig. 2.5. It was built using 3D geometric shapes. Four cuboids of size $50 \times 50 \times 2.5$ cm forming a “T" shape and four pairs of parallel bars enclosing the cuboids have been set over the seabed. This pattern was chosen since it is easy to recognize when the mosaic is built.

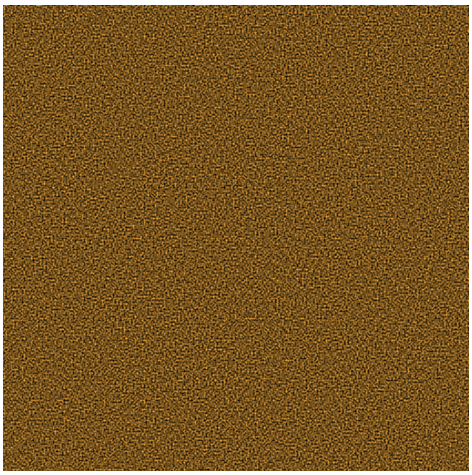
Six seabed textures (see Fig. 2.6) have been set on the bottom, which, when combined with the three trajectories, make a total of 18 data sets. Three of these seabed textures have been produced using random generators: the first one was produced by a Gaussian noise generator with a pixel value independent from its neighbours. Two textures have been produced using a Perlin random process [116, 117], which generates natural appearing textures. An image produced using this method provides a correlation between a pixel and its neighbours that can be seen as smooth transitions. For this work, the Perlin textures are produced with a low and a high correlation. The other three seabed textures have been created by taking photographs from real surfaces. The purpose of using real surface textures is to obtain realistic representations of the seabed on the images.



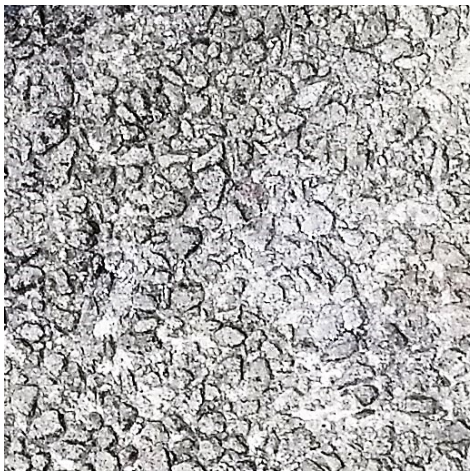
(a) Gaussian noise



(b) Perlin with high correlation



(c) Perlin with low correlation



(d) Photograph of real surface 1

(e) Photograph of real surface 2

(f) Photograph of real surface 3

Fig. 2.6 Seabed textures used in the simulator.

The sensor to be simulated is the DIDSON 300 [2] with FoV of $29^\circ \times 14^\circ$ (aperture and elevation angles, respectively), 96 beams, image size of 512×96 pixels and intensity range between 0 and 255. The images generated by the sonar are truncated in the range to zoom into the objects that appear in the scene only. This interval is chosen because it encloses the area in front of the sonar where the objects are. Noise parameters are set according to measurements from real images in [118]. The Gaussian noise added to the pixels in the image has a mean and standard deviation of 13.7% and 3.1% of the maximum pixel value, respectively, whilst the Rayleigh distribution has a scale parameter of 27.5% of the maximum pixel value.

Fig. 2.7 shows the absolute error of motion estimates, which represents the difference between the estimated and actual displacement between two consecutive images. This was obtained using the Gaussian seabed texture and the sonar trajectory 2.

Table 2.1 summarizes the root mean square error (RMSE) obtained for each of the data sets for the three types of motion. These results show that the trajectory estimation method has a higher performance when the bottom texture presents more independence between pixels, which is the case of the Gaussian noise texture. The image registration process identifies motion of a pixel from one frame to another. However, if a pixel value is very similar to the values of pixels around it and considering that noise has been added, it is possible that the displacement is estimated inaccurately. This problem appears in textures with the Perlin noise, where pixel values have a dependance to their neighbours. Another point to remark is that estimation is more accurate in x translation compared to y translation. This is due to the similarity between y translation and z rotation when pixels move from one point to another between consecutive frames in the polar coordinate system. In this system, when the object pixels change their positions in x direction, they move vertically. However z rotation of objects can be seen as horizontal movement while y translation can be seen as a rotation as it is shown in Fig. 2.8. Therefore, it is more likely that the image registration algorithm interprets the sonar movement in y direction as a z rotation, producing a higher estimation error.

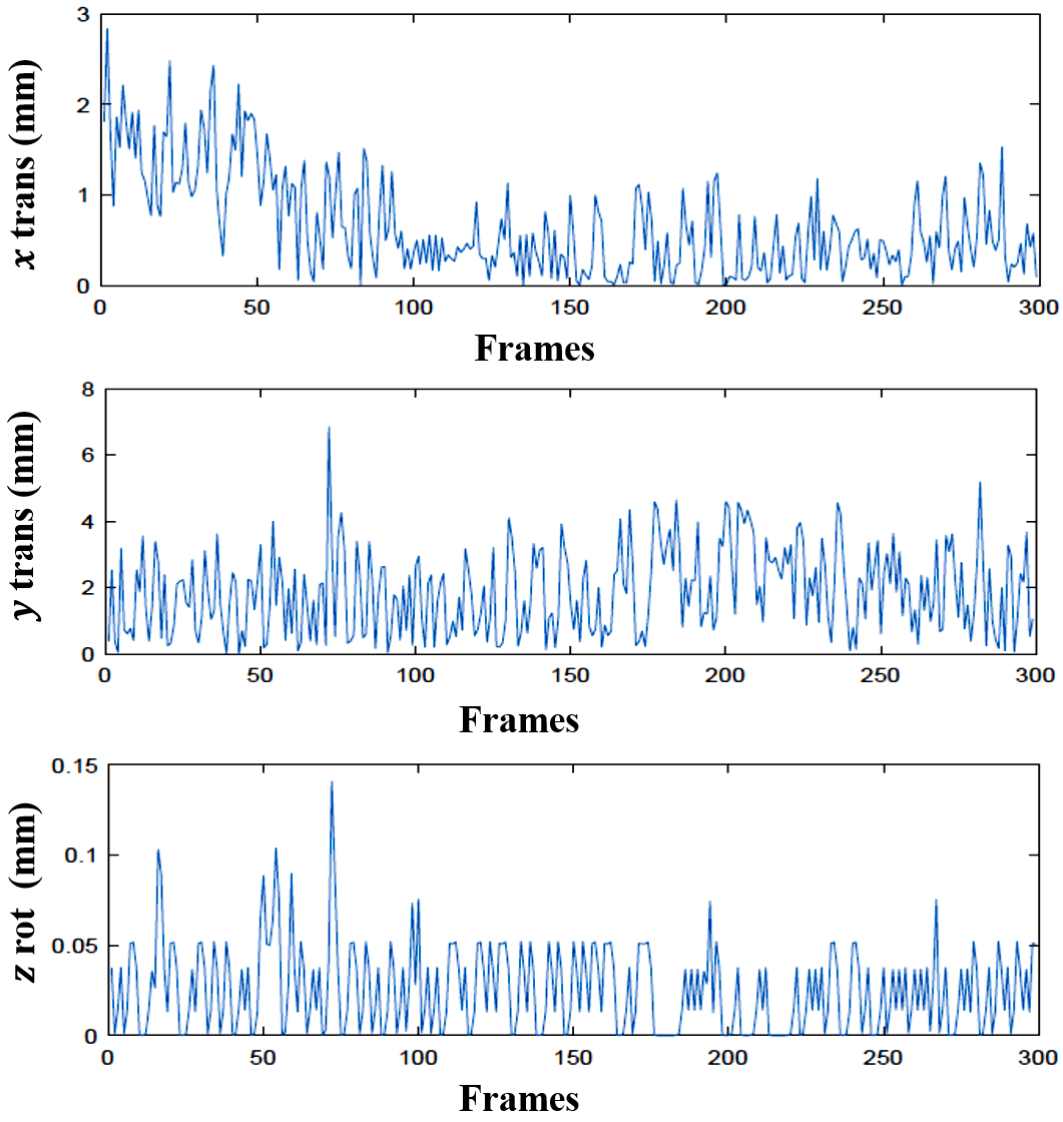
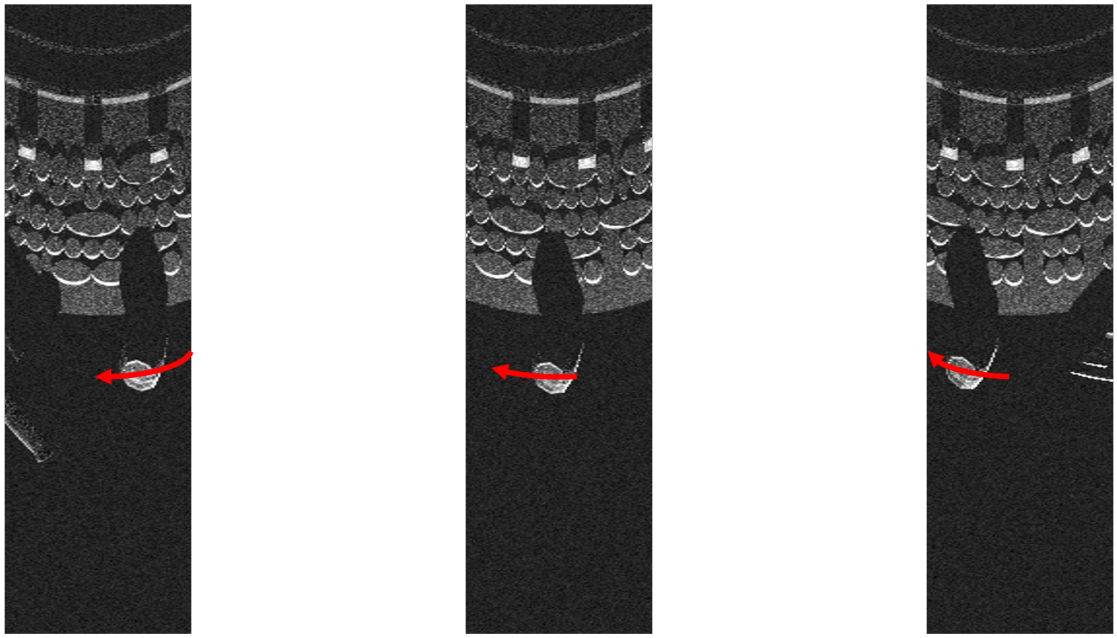


Fig. 2.7 Absolute errors from displacement estimation between consecutive frames from the trajectory 2 data set with Gaussian noise in the seabed texture.

Seabed texture	RMSE of motion estimation								
	Trajectory 1			Trajectory 2			Trajectory 3		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
Gaussian noise	0.95	4.40	1.0	0.86	2.27	3.0	2.88	10.63	31.0
Perlin high correlation	3.33	11.86	6.0	1.70	6.19	8.0	1.13	13.06	41.0
Perlin low correlation	2.45	9.03	4.0	1.28	4.29	6.0	1.17	13.39	38.0
Photograph 1	3.93	12.29	5.0	2.49	5.93	7.0	0.55	11.00	39.0
Photograph 2	4.83	15.98	5.0	4.69	12.59	10.0	0.71	7.55	41.0
Photograph 3	3.95	13.78	5.0	4.24	11.56	9.0	0.78	7.44	41.0

Table 2.1 Errors in the three types of trajectory.



(a) (b) (c)

Fig. 2.8 Sequence of displacement of an object between image frames in polar coordinates. When the sonar moves in y direction, the objects in the images seem to rotate rather than displacing horizontally.

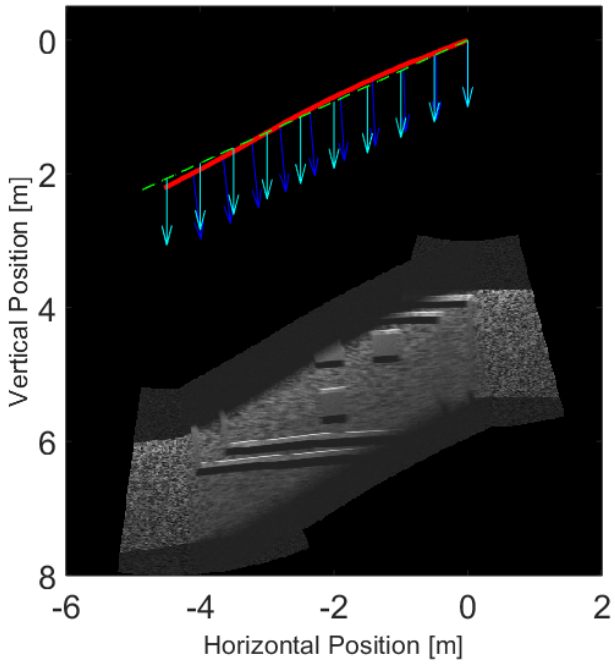


Fig. 2.9 Mosaicing of a data set of 300 simulated frames. The red and green lines represent estimated and simulated trajectories, respectively. Dark blue and cyan arrows represent the attitude of the sonar for estimated and simulated data, respectively.

A mosaic is built when the attitude-trajectory estimation is completed. Fig. 2.9 shows an example of such a mosaic for the data set with the Gaussian seabed texture and trajectory 2.

2.4 Conclusions

The imaging sonar simulator presented in this chapter is able to generate realistic data sets of images from 3D scenarios, exploiting capabilities of the Unity platform. Sonar features such as FoV, maximum range and number of beams can be adjusted in accordance with real sensors. A large number of different imaging sonars can be simulated using this tool as long as their parameters are known and set in the simulator. Also, other noise models can be implemented and customized for different sonars. Attitude and position parameters of the sonar are provided by the simulator, allowing the simulated images to represent a ground truth, e.g., for image registration techniques. As an example, when the images are used as

data source for the attitude-trajectory estimation and mosaicing, it was possible to make a quantitative analysis of this method.

Chapter 3

DL architectures for navigation using sonar images

In this chapter, the use of supervised DL networks to process sonar images for underwater navigation is investigated. State-of-the-art DL networks for micro-navigation using sequences of optical images have been adapted to work with sonar images and along with a new proposed architecture, they have been used for motion estimation. The DL networks are trained using images generated by the simulator presented in Chapter 2. The data sets are made using pairs of consecutive images associated with labels that represent the motion of the sonar between images. Section 3.1 summarizes the process of using DL networks trained with simulated images for motion estimation of underwater vehicles. In Section 3.2, the data sets and sonar parameters employed for training and validating the networks are described. Section 3.3 describes the DL architectures used for motion and trajectory estimation and specifies the modifications made to work with the sonar images. Section 3.4 presents a performance comparison of the DL architectures. Section 3.5 describes the procedures and presents results of trajectory estimation and mosaic building using synthetic and real sonar images. In Section 3.6, the performance improvement when using multiple sonar sensors compared to a single sensor is investigated. Conclusions are given in Section 3.7.

The work in this chapter is presented in the papers: J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, “Deep learning architectures for navigation using forward looking sonar

images," *IEEE Access*, vol. 9, pp. 33 880–33 896, 2021; J. E. Almanza-Medina, B. Henson, and Y. Zakharov, "Motion estimation of an underwater platform using images from two sonar sensors," *TechRxiv preprint techrxiv.15059991.v1*, 2021

3.1 Introduction

DL techniques for image registration have been developed for such applications as optical flow and ego-motion estimation [69, 76, 119–121], displacement in magnetic resonance images [122, 123] and synthetic aperture radar images [124–126]. As mentioned in Chapter 1, for underwater scenarios, scarce effort has been dedicated on the use of DL techniques for navigation using acoustic imagery. In this chapter, a framework that uses advanced DL architectures for real-time motion estimation from consecutive sonar images is implemented.

The DL architectures are trained using images generated by a sonar simulator. Then, the estimates are used to reconstruct the navigation trajectory of the vehicle. Additionally, a mosaic that combines the sonar images is built following the reconstructed trajectory. The implemented DL architectures are based on DL architectures developed for motion estimation using optical images. Analysis of the state-of-the-art techniques from the literature shows that the following architectures are sufficiently advanced to achieve high precision motion estimation: SfMNet [119], PoseNet [69, 120], CNN1b and CNN4b networks [121]. Different from optical images, sonar images are monochrome and have lower resolution [34], making them less informative for the motion estimation. These architectures are modified for working with sonar images and to investigate the performance of the motion estimation.

From the machine learning point of view, this is a regression problem. When training a neural network for complex regression tasks, a large amount of labeled training data is required [127], e.g., tens of thousands images for optical flow estimation [128, 129]. Manual labeling of training data can be a time consuming task [130]. Furthermore, in some real scenarios, it can be hard to obtain adequate data to train a network [131]. Data augmentation can partly resolve this problem by artificially enlarging the size of the training set whilst keeping the labels [132]. The use of synthetic data generated by computer can

be an alternative to alleviate this issue and can be used as a source of large sets of labeled training data [127].

Synthetic data sets made entirely by computers have been used in DL for training and testing purposes, specifically in computer vision applications such as face recognition [133, 134], object detection/classification [131, 135, 136, 130], text detection and recognition [137, 138], captcha recognition [127], etc.

The FLS simulator from Chapter 2 is used to generate large volumes of synthetic sonar images. The images are used for training and validation of the DL networks. The process followed for training the networks is illustrated in Fig. 3.1a and it consists of the following steps:

1. The sonar simulator generates images while moving in simulated environments and stores the sonar position where each image is generated.

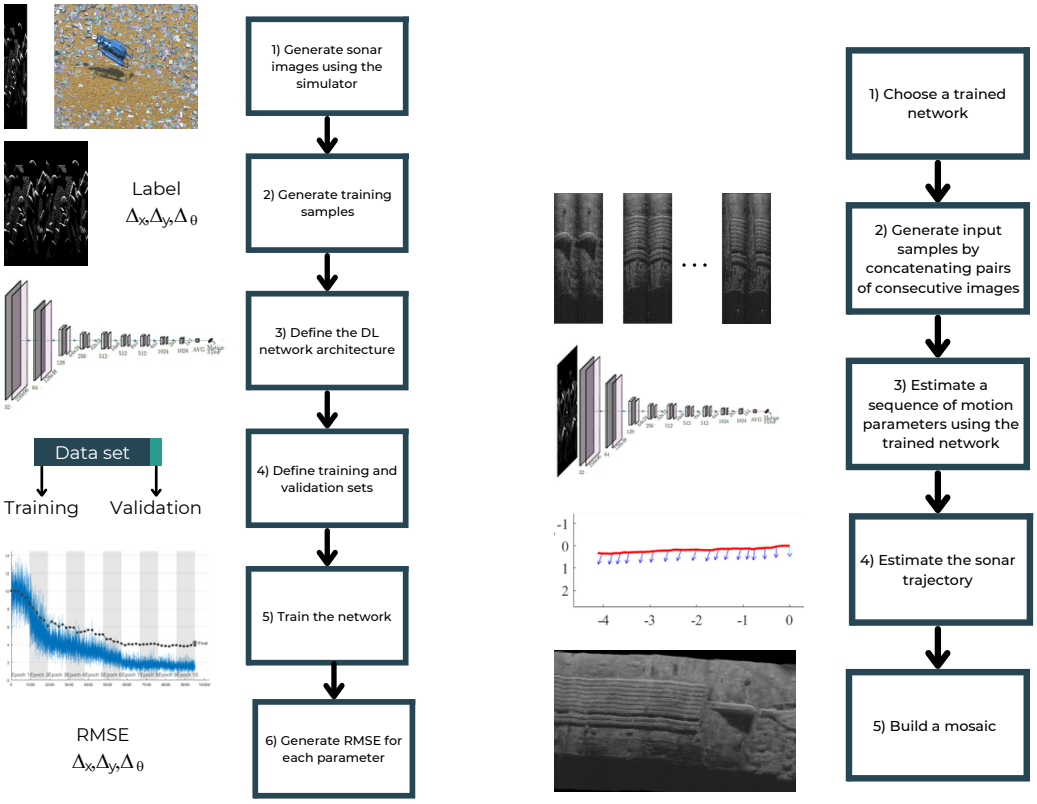
2. Data sets for training and validation are generated using the images and positions. Each training sample is obtained by concatenating a pair of consecutive images into one single image. The label corresponds to three motion parameters required to move from the position where the first image is acquired to the position where the second image is acquired as detailed in Section 3.3. The labels are quantized and normalized as described in subsections 3.4.1 and 3.4.2, respectively. A data sample consists of a concatenated image and a label.

3. The DL network architecture is defined. Five different architectures are considered as described in Section 3.3.

4. The data set is split into a training set and a validation set (95% and 5% of the whole data set, respectively). For a fair comparison, each network is trained using the same data.

5. The network is trained by using the training samples as inputs.

6. Once the network is trained, the RMSE for each of the three parameters is calculated.



(a) (b)

Fig. 3.1 (a) Data flow diagram of the process for training the DL networks with data generated by the simulator. (b) Data flow diagram of the process for motion and trajectory estimation using a trained network.

The trained networks are used to estimate the motion of the sonar vehicle from a simulated or real data set obtained along a trajectory. This process is illustrated in Fig. 3.1b and described as follows:

1. An already trained network is chosen to estimate the motion parameters from sonar images in the data set.

2. Pairs of consecutive images in the data set are concatenated into a single image. There are no training labels since the network is used for estimation.

3. The concatenated images are applied to the network, which produces estimates of three motion parameters for every concatenated image.

4. The trajectory of the sonar is computed as described in subsection 3.5.1.

5. Optionally, a mosaic can be built by merging the sonar images according to the estimated trajectory.

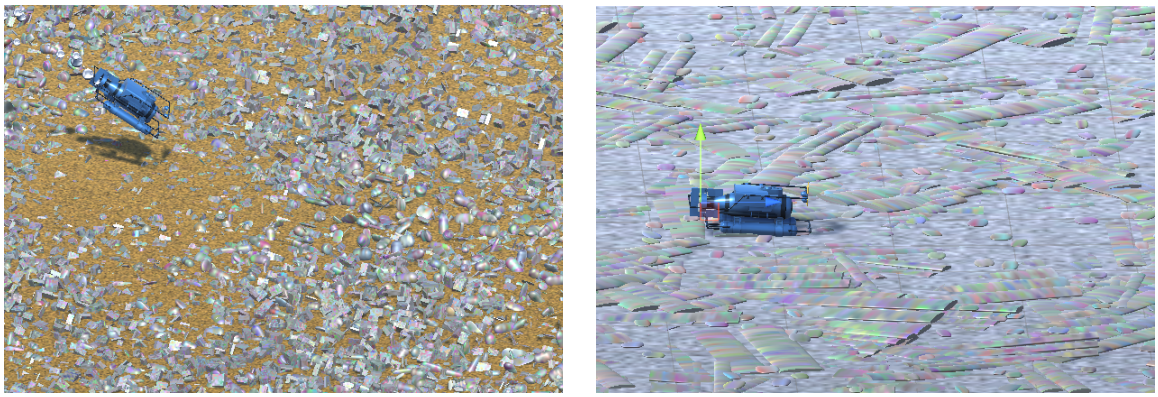
3.2 Sonar application scenarios and data sets

In this section, four sonar image data sets are described. Two of these data sets are used for training and validating the network. They are described in detail in subsections 3.2.1 and 3.2.2. Both data sets are made of synthetic images generated using the simulator from Chapter 2. Examples of simulation scenarios used for each data set are shown in Fig. 3.2. The first one represents an underwater bottom field covered with rocks, whilst the second one represents the surface of a ship's hull (Fig. 3.2a and 3.2b, respectively).

When generating data from a simulator, there always is a modeling error, resulting in a reduced accuracy of the estimator. The simulator parameters should be chosen to guarantee a minimum of this error. With the sonar simulator, such parameters as the level of noise, size and reflectivity of objects in the simulated environments should be adjusted to match the real scenarios where the estimator is going to be used. The accuracy of the estimator can be improved using training and validation with a combination of simulated and real data. However, labeling the real data is a complicated problem. The accuracy in real scenarios can also be improved with sufficiently high variability of the simulated data. The ultimate test of an estimator should be preferably done using real data. The sonar simulator is used to generate a highly variable data set for training the networks and further test the motion estimator using real data.

The simulated sensor is the DIDSON 300 sonar (FoV of $29^\circ \times 14^\circ$ in azimuth and elevation angles, respectively, 96 beams in the azimuth dimension, image size of 512×96 pixels and intensity range between 0 and 255). The frame rate is 21 frames (images) per second (fps). The images are generated without image noise, which is added later for validation and network performance enhancement purposes.

Two more data sets are described in subsections 3.2.3 and 3.2.4. They consist of images acquired from real sonar sensors in the inspection of a ship's hull and a dam wall, respectively.



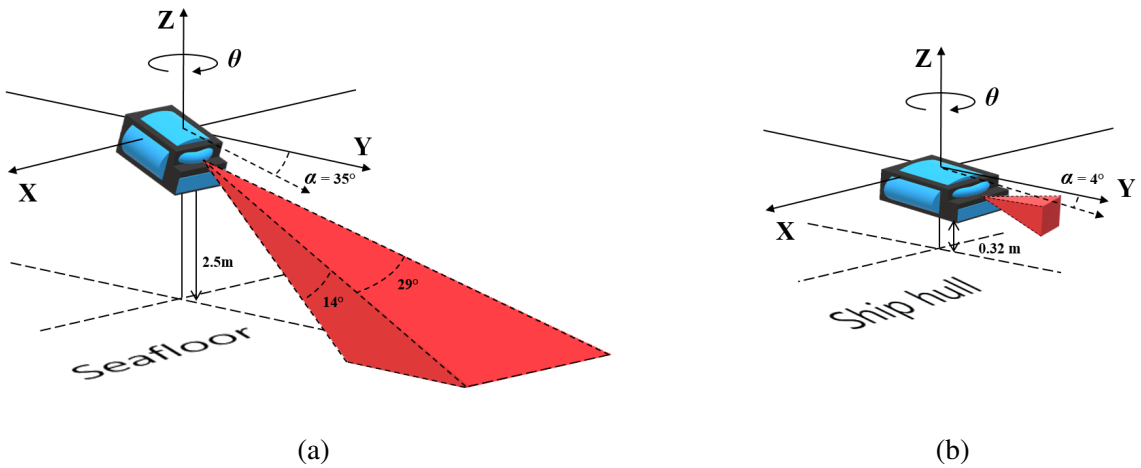
(a) (b)

Fig. 3.2 (a) Simulated rocky bottom field underwater scenario. (b) Simulated ship's hull surface scenario.

3.2.1 Simulated rocky field data set

For the rocky field data set, thirty different scenarios have been created and 200,000 images generated. Each scenario has a flat seabed, where three different types of geometrical objects are placed to simulate rocks: cubes, capsules and cylinders. Ten scenarios have an area of 30×30 m and twenty scenarios have an area of 50×50 m. From each of the first ten scenarios, 5000 images have been generated and from each of the other twenty scenarios 7500 images have been generated.

The difference between scenarios lies in the position and the number of elements of each type of rock. The number of elements is defined by a square grid of size p by side, where p is an integer number in the range from 30 to 130. A rock is placed on each vertex of the grid. The value of p is different for the cubes, cylinders and capsules. Therefore, there are 3 different values of p for each scenario. Also, p values change from scenario to scenario. Each rock is slightly shifted from its corresponding vertex by values ξ_x and ξ_y on the xy-plane of the grid, respectively. Both values are randomly generated for each single rock using a uniform distribution within the grid step. The height from the seafloor is a random value between 0 and 0.45 m. The size of a rock in each of the three dimensions is a random value uniformly distributed in the range from 0 to 0.45 m. The rocks are rotated around each axis



(a) (b)

Fig. 3.3 Coordinate system of the sonar vehicle used for generating the data sets. Forward/backward motion of the vehicle corresponds to the y-axis, whilst sideway motion corresponds to x-axis. The height from the seafloor is constant. Rotation around the z-axis corresponds to the parameter θ. (a) Sonar height and pitch (α) of 2.5 m and 35°, respectively, used in the rocky field scenarios. (b) Sonar height and pitch of 0.32 m and 0.4°, respectively, used in ship's hull scenarios.

by a random number in the range of -165° and 100°. The rotation value is independent for each axis.

Considering that in the simulator the acoustic reflectivity of the objects is coded by a number between 0 and 1, the reflectivity is generated randomly by dividing the whole seabed into a grid of 2048 by 2048 cells, where each of the grid cells has a random value chosen from a uniform distribution. For the rocks, the reflectivity on their surface is assigned according to an image generated using the Perlin algorithm for creating realistic random images [117].

The simulated sonar is looking down with a rotation angle around x-axis (α) of 35° (see Fig. 3.3a). The height from the seafloor is fixed at 2.5 m. The simulator generates a large number of sequences of 5 images. For each sequence the sonar moves at a constant rate in each DoF; the rate is defined randomly with a uniform distribution within limits of the maximum motion and rotation speed of the sonar vehicle. The limits for x and y-translations are $\pm$ 420 mm/s and the limit for rotation around the z-axis is $\pm$ 9.45°/s (the maximal displacement of $\pm$ 20 mm and rotation of $\pm$ 0.45° between consecutive images, respectively). The maximum translation value is selected according to the specification of

the Bluefin Robotics Hovering Autonomous Underwater Vehicle [139] and an approximation to the observed translation in the real data set described in subsection 3.2.3. For the rotation, the maximum speed was chosen focusing on low rotation navigation, such as ship's hull inspection.

After generating a sequence of 5 images, the sonar is randomly moved to a different place in the simulated scene. It is displaced within 3 to 4 m in each of the x and y-axis, while the rotation around the z-axis is a uniform random value in the range between $-$ 180 and 180°. This large displacement is to avoid over-generating images of the same view and area. After moving to a new place of the scenario, the next sequence of 5 images is generated using new motion rates. In total 40,000 sequences of five images have been created. The data set is made by concatenating pairs of consecutive images from each sequence. Therefore 4 pairs are made from each sequence. The total number of pairs of images in the data set is 160,000.

The data set labels correspond to the displacement of the sonar vehicle from the position where an image is obtained to the position of the next image. Each label has three parameters that correspond to x- and y-translation and rotation around z-axis. The data set was shuffled and divided into training and validation sets with 95$\%$ and 5$\%$ of the data, respectively.

3.2.2 Simulated ship's hull data set

Fifteen scenarios have been created to simulate the bottom of a ship's hull. In each scenario, a flat surface with two different types of objects is simulated: (i) groups of flat tubes aligned one beside the other and (ii) small flat cylinders that represent sacrificial anodes. The objects are attached to the ship's hull surface. They can be seen in Fig. 3.2b. The tubes are generated as elliptical cylinders. All the sizes are independent uniformly distributed random variables in the following ranges. The cross-section of a tube is an ellipse with the major axis in the range from 0 m to 0.4 m and the minor axis in the range from 0 m to 0.05 m. The length of a tube is between 1 m and 3 m. The anodes are also elliptical cylinders. Their cross-section is facing up. The height is in the range from 0 m to 0.03 m. The major and minor axis are in the range from 0.15 m to 0.3 m and from 0.05 m to 0.1 m, respectively. The position of an object is generated as in the rocky field scenarios. The hull surface reflectivity is generated

by dividing the whole surface into a grid of 1024 $\times$ 1024 cells. A random value is assigned to each cell using a uniform distribution between 0.25 and 1. The reflectivity on the surface of the tubes and anodes is defined with the Perlin algorithm. The simulated sonar height from the hull and pitch angle (α) are fixed at 0.32 m and 4°, respectively (see Fig. 3.3b). The height and pitch are chosen based on estimates from [79] for a real ship's hull data set. The sonar translation and rotation speed limits are the same as that in the rocky field scenarios. From the fifteen scenarios, a total of 100,000 images have been generated. Then from each sequence of 5 images, 4 pairs of images are made to make a total of 80,000 pairs of images in this data set.

3.2.3 Real ship's hull data set

This data set was obtained using a Bluefin Robotics Hovering Autonomous Underwater Vehicle equipped with a DIDSON 300 sonar as described in [140]. The data set consists of a sequence of 4464 images that show the inspection of a ship's hull. A subsequence of 520 images is extracted. The subsequence represents one single pass from end to end of the ship, for a total length of around 10 meters. The sonar generates 21 images per second (21 fps).

3.2.4 Real dam inspection data set

This data set is from the inspection of a dam wall [3, 4]. It consists of 1596 images obtained using the ARIS 3000 sonar [141] while doing a single pass along the dam wall, moving principally in the sideways direction.

3.3 DL networks for attitude-trajectory estimation

In this section, five DL networks for trajectory estimation are described. The first four networks are state-of-the-art networks for trajectory estimation using optical images and the fifth one is a new architecture.

Six DoF are needed to represent the motion of a sonar sensor. They correspond to the translations in x-, y- and z-axes and the rotation around each axis. The height from

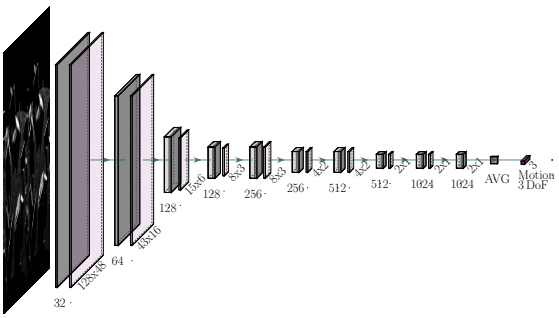
the seafloor is assumed to be constant for the duration of the sonar exploration and that rotations around x- and y-axes are negligible. Therefore, the DL networks have been adapted to work with sonar images for motion estimation between a pair of images in three DoF: $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]$, translation on the xy-plane parallel to the seabed and rotation around the z-axis (denoted by θ), respectively (see Fig. 3.3).

The network architectures are shown in Fig. 3.4. The input of all the networks is a data set of images obtained by concatenation of a pair of images. Since each image size is 512×96 pixels, the size of the concatenated input image is 512×192 pixels.

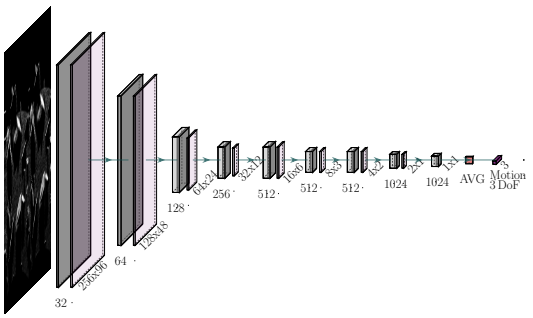
3.3.1 SfMNet

SfMNet [119] is a self-supervised DL network designed to estimate the camera motion from a sequence of images (rotation and translations). The architecture of the SfMNet is divided into two sections: the first section uses convolutional layers to estimate the camera motion and the second section uses deconvolutional layers to obtain a motion mask that is used to generate a pixel motion estimate. From the whole architecture, only the first section is implemented since it provides the motion estimation. The network architecture is shown in Fig. 3.4a. The architecture from [119] is repeated as much as possible but some modifications are required to deal with the sonar images. Ten convolutional layers are used to build the network; only one of the 64 channels layer from the network in [119] is used and the other is removed. The network has shown better results and lower training time without this layer.

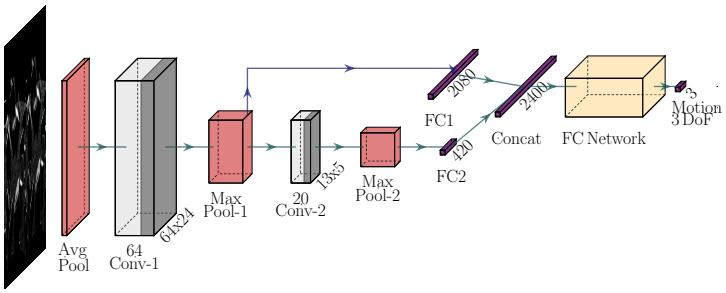
The kernel size of the convolutional layers is 3×3, except for the first two layers, where the kernel size is 7×7 and 5×5, respectively. The stride of the first three layers is 4, 3 and 3, respectively. The remainder of the convolutional layers have a stride that alternates between 2 and 1 in each layer and the number of channels increases by two every two layers. A batch normalization layer and a rectified linear unit (ReLU) activation function are placed after each convolutional layer, except for the last one that is connected to an average pooling layer. The ReLU output is equal to 0 when the input is less than zero, otherwise the output is equal to the input ($g(x) = \max(0, x)$) [142]. The average pooling layer improves the performance when images are noisy. The final layer is a regression layer that outputs the 3 estimated DoF.



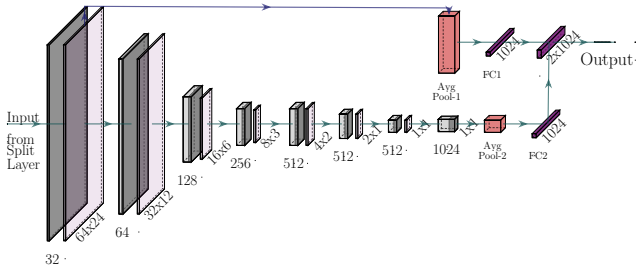
(a) SfMNet - Motion estimation section



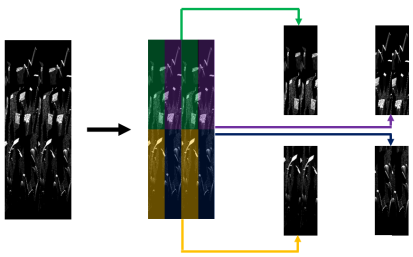
(b) GeoNet - PoseNet



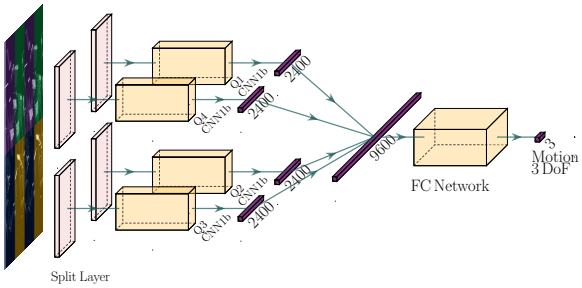
(c) CNN-1b



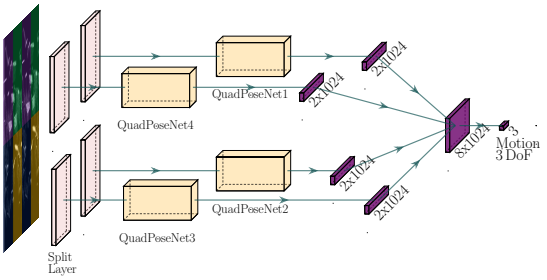
(d) QuadPoseNet



(e) Split layer



(f) CNN-4b



(g) CNN-4QuadPoseNet

Fig. 3.4 DL architectures for estimation of three DoF of position and orientation of an underwater vehicle using acoustic images.

Dropout regularization with a rate of 50% is applied when training the network to prevent overfitting.

3.3.2 GeoNet: PoseNet

GeoNet [120] is also a self-supervised DL architecture that can perform ego-motion estimation from videos. The full network architecture consists of three subnetworks called DepthNet, PoseNet and ResFlowNet. For the work in this chapter, only the PoseNet architecture (position and orientation estimation) is of interest. The PoseNet is based on the architecture from [69]. This network receives a sequence of Q images as input, then 7 convolution layers with stride 2 are applied, followed by a 1×1 convolution layer with $6 \times (Q-1)$ channels, six DoF are estimated by comparing one of the Q images with the others. All the convolutional layers, except the last one, are followed by a ReLU activation function and batch normalization. An average pooling layer is applied before the output regression layer to generate the motion estimates. This architecture was implemented and trained. However it did not show any learning progress when applied to the sonar images. Therefore it was modified by adding extra convolutional layers to make a total of 9 such layers as shown in Fig. 3.4b, where the number of channels in each convolutional layer can be seen below each block. Dropout regularization with a rate of 50% is applied during the training. The output regression layer size is $3 \times (Q-1)$, since it estimates three DoF. The number of input images is $Q = 2$.

3.3.3 CNN-1b and CNN-4b

In [121], a DL framework with three types of CNNs to determine the trajectory of a camera from multiple images is presented. In this work, two of the three CNNs are implemented and compared: CNN-1b and CNN-4b. The network parameters of the CNN-1b are adjusted to work with sonar images. The images are smaller than the optical images used in this reference. Therefore, the internal layers of the network are adjusted accordingly. The adjusted CNN1-b can be seen in Fig. 3.4c. It takes an image and downsamples it with an average pooling layer with kernel of 8×8. It reduces the number of parameters to train, and, as a result, the computational cost. After downsampling, the layers are as follows: a convolutional kernel of size 9×9, a 4×4 max pooling layer, another convolutional kernel of size 2×2 and a 2×2

max pooling layer. The outputs of each max pooling layer are concatenated into one single fully connected layer that is then connected to a fully connected network, made of 4 fully connected layers to obtain the motion estimate.

The parameters of the network CNN-4b are adjusted to to match the size of the sonar images. The adjusted network is shown in Fig. 3.4f. It uses a Split layer (Fig. 3.4e) to segment each of the concatenated images into 4 quadrants, then the corresponding quadrants are concatenated to make a sub-image. Each sub-image is downsampled 4 times by an average pooling layer. CNN-1b is then applied on each sub-image. The final layer combines the outputs of the four CNN-1b networks. The fully connected network takes the outputs of the concatenated layers as the input. The sizes of the fully connected layers are 2400, 1200, 600 and 300, respectively. The final layer of the network is the regression layer.

3.3.4 CNN-4PoseNet

A new architecture is proposed to exploit the segmentation into quadrants implemented in the CNN-4b network and the high number of layers of GeoNet-PoseNet. The complete network is displayed in Fig. 3.4g. The input of the network is the pair of concatenated images ($Q = 2$). Similarly to the CNN-4b network, the images are split into 4 sub-images and the quadrant of one image is concatenated with the corresponding quadrant of the other image. Then each of the 4 concatenations is passed to a subnetwork called QuadPoseNet. This subnetwork consists of 8 convolutional layers with a ReLU activation function and a batch normalization layer. The convolutional layers use the same kernel size of the convolutional layers as the PoseNet described above. After the last convolutional layer, an 4×4 average pooling layer is used, whose output applies to a fully connected layer of size 1024. Additionally, the output of the first ReLU function after the first convolutional layer is connected to a 5×5 average pooling layer. Then this layer is connected to another fully connected layer of size 1024. The two fully connected layers of size 1024 are combined into one single fully connected layer of size 2×1024 to make the output of each QuadPoseNet. Combining layers from the beginning and end of the series of convolutional layers follows the approach of the original CNN-4b to combine coarse and fine features from the images to perform the estimates. The

QuadPoseNet layers can be seen in Fig. 3.4d. The outputs of the four QuadPoseNets are concatenated into another fully connected layer of size 8×1024 and then the final regression layer to compute the motion estimates.

3.4 Performance analysis of the DL networks

3.4.1 Motion estimation using simulated data

The pairs of images in polar coordinates are the input to the networks. The labels for displacement Δ_x and Δ_y are given by the simulator in metres and in degrees for the rotation label Δ_θ. The three parameter labels are transformed into parameters measured in the same units by a quantization process using the minimum and maximum values of each type of label. Five bits of resolution are used to define the quantization levels, i.e., the labels are assigned to one of 32 values (0 to 31). The quantized labels are defined by $\Gamma = [\Gamma_x, \Gamma_y, \Gamma_\theta]$.

A regression output layer with the mean squared error (MSE) loss function is used to measure the estimation error during the training. The loss function $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2SW}\sum_{k=1}^{S}\|\widehat{\Gamma}_k - \Gamma_k\|^2, \tag{3.1}$$

where $\widehat{\Gamma}_k = \left[\widehat{\Gamma}_{x_k}, \widehat{\Gamma}_{y_k}, \widehat{\Gamma}_{\theta_k}\right]$ are the estimates for each DoF, k is the index that refers to the training examples in the mini-batch, S is the mini-batch size and W is the number of parameters to estimate, which is $W = 3$.

To speed up the training, the data set is split into mini-batches ($S = 4$). The learning rate at the start of the training is set to 0.0001 and it is reduced after 12 epochs. The networks are trained until the validation loss converges. This takes less than 32 epochs depending on the network. The Adam optimization algorithm is used for training [143].

The five networks are trained using noiseless images. The validation is performed using the validation set with noise and without noise. Noise with parameters measured from real images in [118] are applied to the pixel intensity levels. The pixels that correspond to the acoustic shadows in the images are distorted by additive Gaussian noise, with the mean and a

standard deviation of 13.7% and 3.1% of the maximum pixel value, respectively. Pixels of objects are modified using an additive noise with the Rayleigh distribution that represents the scattering noise from the surface of the objects. The Rayleigh distribution has a scale parameter of 13.7% of the maximum pixel value.

The validation results for all the architectures are shown in Table 3.1. It can be seen that the architectures with the best performance, when applied to noiseless images, are the PoseNet and the proposed CNN-4PoseNet, achieving a similar navigation RMSE in translation in x-direction and rotation around z-axis. The RMSE in translation in x-direction is slightly smaller for PoseNet whilst in translation in y-direction, it is smaller for CNN-4PoseNet. Validation using noisy images shows that the proposed CNN-4PoseNet is less sensitive to the noise than the other networks. CNN-1b and CNN-4b architectures are the most sensitive to the noise.

Also, it can be seen that all the architectures achieve better estimates in the y-axis, corresponding to the forward/backward motion of the vehicle. When working with images produced by sonars with a small azimuth FoV, it is more difficult to estimate the sideway motion of the vehicle since rotation around z-axis and x-translation result in very similar distortions in images. The forward/backward movement of the sonar means that the pixel motion is mostly in the range axis, thus making it largely independent of the pixel motion caused by sideway movement and rotation around the z-axis.

To measure the similarity of the estimated parameters, the cross-correlation of the estimation errors is calculated. The PoseNet estimates are used to compute the cross-correlation. The cross-correlation values obtained are ρ_{yx}=0.025, $\rho_{y\theta}$=0.220 and $\rho_{x\theta}$=0.822, for the cross-correlation between translation motions, the motion in y-direction and the rotation and the motion in x-direction and the rotation, respectively. From these values, it can be seen a high correlation between errors obtained for estimates of the sideway movement of the sonar and its z-rotation.

The SfMNet, PoseNet and CNN-4PoseNet architectures are selected for further modifications to try to further reduce the RMSE of motion estimation. They are chosen since

Approach	RMSE of motion estimation					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
SfMNet	4.86	2.36	8.4	5.43	3.05	9.8
PoseNet	2.81	1.44	4.9	11.32	6.42	19.9
CNN-1b	7.63	4.63	17.2	13.67	10.60	26.6
CNN-4b	4.22	2.27	8.9	12.41	13.26	29.1
CNN-4PoseNet	3.18	1.15	5.0	4.87	2.63	8.3

Table 3.1 Validation errors of the networks.

they achieved the lowest RMSE among the architectures. The following subsections present results of these attempts.

3.4.2 Normalization of the labels

Normalization is applied to the labels of the training and validation data, rather than using the quantization. The labels are normalized to their maximum values, defined by the limits to the displacement and rotation. It was observed that when normalizing the motion parameters to the range from -1 to 1, the networks are not capable of learning during the training. However, when normalizing the motion parameters to lie in the range from -10 to 10, the networks can learn and obtain better estimates than the quantized-labels approach for some parameters. The benefit of using the range from -10 to 10 can be related to the magnitude of the weight values when the internal layers of the networks are randomly initialized.

A comparison between the quantization, and normalization $\times 10$ is presented in Table 3.2. For PoseNet, the validation using noiseless images shows better estimates for y-translation when using the normalization and almost the same estimation errors for x-translation and z-rotation. For noisy images, a large improvement can be seen in the estimates when using the normalization approach. This makes the normalization of labels more preferable than the quantization.

The SfMNet with the normalization achieves better estimates than with the quantization for all the parameters when images are without noise. For noisy images, the normalization results in slightly higher estimation errors when compared with the quantization.

For CNN-4PoseNet, the normalization results in increasing all the estimation errors.

From the results obtained, the PoseNet with normalization $\times 10$ (PoseNet-Normx10) and CNN-4PoseNet with quantization (CNN-4PoseNet-Qua) are selected to apply further modifications in their training strategy to reduce the estimation error.

Approach	**RMSE of motion estimation**					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
SfMNet-Qua	4.86	2.36	8.4	5.43	3.05	9.8
SfMNet-Normx10	4.16	2.14	7.3	5.55	3.47	10.6
PoseNet-Qua	2.81	1.44	4.9	11.32	6.42	19.9
PoseNet-Normx10	2.84	1.00	5.2	6.54	3.39	17.0
CNN-4PoseNet-Qua	3.18	1.15	5.0	4.87	2.63	8.3
CNN-4PoseNet-Normx10	5.35	1.72	7.9	10.21	4.23	23.9

Table 3.2 Validation errors for SfMNet, PoseNet and CNN-4PoseNet with quantization and normalization.

3.4.3 Training with noisy images

The rocky data set images are modified by adding a low level of noise and they are then used for training the PoseNet-Normx10. The noise added to the images is Gaussian with a mean of 10.2 and standard deviation of 5.1, which correspond to 4% and 2% of the maximum value of intensity of a pixel (255), respectively. The percentages are chosen to alter the images with a low level of noise only. This Gaussian noise is applied to pixels in acoustic shadows. As described in Chapter 2, the noise applied to pixels of the objects in the images has a Rayleigh distribution. The scale parameter of the distribution is 10.2 (4% of the maximum value of a pixel intensity). This approach is validated using noiseless images and images with the higher level of noise as described in subsection 3.4.1.

The results presented in Table 3.3 show that for both, CNN-4PoseNet-Qua and PoseNet-Normx10, training without noise is slightly better for noiseless images. When validating with noisy images, the PoseNet-Normx10 shows better estimates by the network trained with noisy images. The CNN-4PoseNet-Qua trained with noisy images obtains similar results to training with noiseless images. The results from PoseNet-Normx10 suggest that the level of noise in training images should be considered when the networks are applied to real data.

Based on the results obtained when training with quantization, normalization and using noisy images, the PoseNet-Normx10 is selected to continue with further modifications to reduce the estimation error. The CNN-4PoseNet is discarded given that it shows smaller improvement. Also, the CNN-4PoseNet is less suitable for a real-time implementation since its training and testing times are between 2 and 3 times higher compared to the PoseNet.

Approach	**RMSE of motion estimation**					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
PoseNet-Normx10	2.84	1.00	5.2	6.54	3.39	17.0
PoseNet-Normx10wNoise	2.94	1.26	5.6	3.63	1.50	7.0
CNN-4PoseNet-Qua	3.18	1.15	5.0	4.87	2.63	8.3
CNN-4PoseNet-QuaNoise	5.00	2.66	8.1	5.71	1.61	6.5

Table 3.3 Validation errors when training the PoseNet-Normx10 using images with noise and without noise.

3.4.4 Concatenation of 3 images

Rather than using two concatenated images (Q = 2) as input, 3 images are concatenated (Q = 3) and used for training the PoseNet-Normx10. The basis of this is to give more information to the network and thus reduce the estimation error. From each sequence of 5 images from the rocky data set, 3 training samples are generated by concatenating 3 consecutive images, resulting in a total of 90,000 samples. The label is the 3 DoF of motion of the vehicle that corresponds to a constant motion rate for each sequence. The results are shown in Table 3.4. The 3-image approach has an improvement in the estimation accuracy

and higher robustness to noise in images in both axes for translation motion compared to the 2-image approach. However the error in rotation estimation when using noisy images is increased.

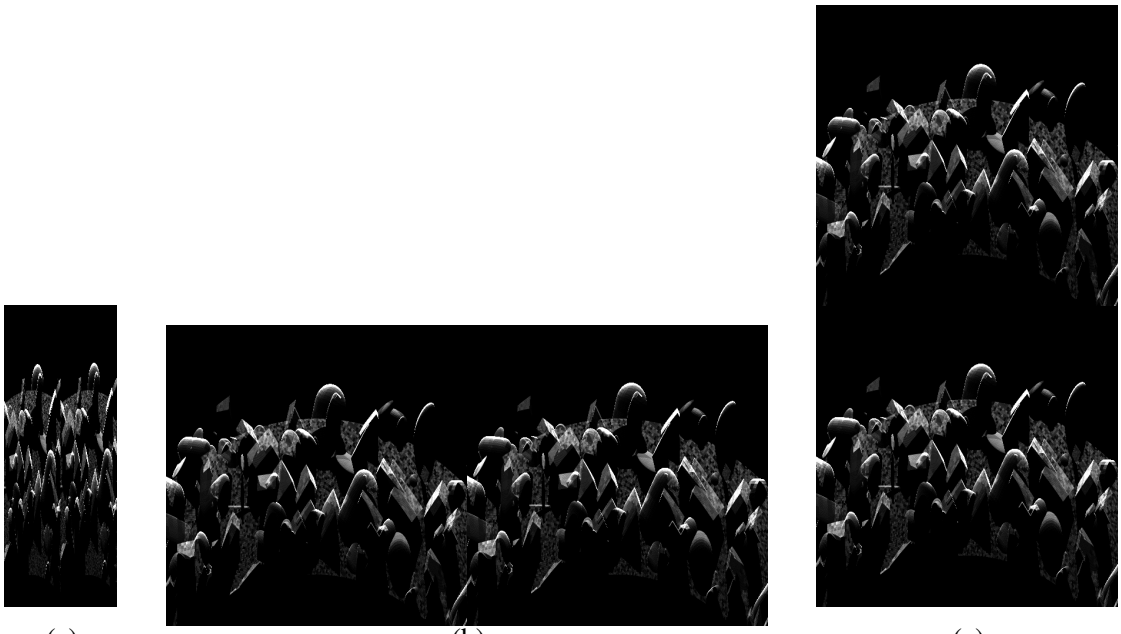
Approach	**RMSE of motion estimation**					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
Using 2 images	2.84	1.00	5.2	6.54	3.39	17.0
Using 3 images	2.70	0.87	4.6	4.96	1.38	22.7

Table 3.4 Validation errors for concatenation of 2 and 3 images in PoseNet-Normx10.

3.4.5 Using wider images

The sonar parameters are modified to have 512 beams and a FoV of $60° \times 12°$ (azimuth and elevation angles, respectively). Having more beams and therefore more information in the sideway motion is expected to reduce the estimation error for motion along x-axis and z-rotation. A small data set of 40,000 pairs of images was generated. The images are obtained using the first 10 rocky field scenarios of 5000 images each. To generate the images, the sonar follows the same positions as for generation of the original rocky field data set. Two types of concatenation of images are considered, vertical and horizontal, i.e., one image beside the other and one image above the other, respectively. The purpose of this is to investigate if the network can achieve a better result depending on the way that two images are concatenated. The labels are set according to the movement of the sonar between images as before. For visualization, examples of pairs of concatenated images for the original data set (29°/96 beams) and the horizontal/vertical concatenated wider images data set (60°/512 beams) are presented in Fig. 3.5.

The network used for training is the PoseNet-Normx10. The results obtained can be seen in Table 3.5. For a fair comparison when using the original images, the network was re-trained using only the first 40000 pairs of the data set. The results show that wider images provide slightly better sideway estimation with noiseless images than the original ones but



(a) (b) (c)

Fig. 3.5 Concatenated pairs of sonar images with different azimuth FoV, number of beams and type of concatenation. (a) 29°/96 beams - horizontal concatenation. (b) 60°/512 beams - horizontal concatenation. (c) 60°/512 beams - vertical concatenation.

there is an increase in the error for the forward/backward motion estimation. The error is higher compared to the original network when validating with noisy images. The higher errors for the wider images with noise can be due to the fact that wider images have a larger area with black pixels (with no information). Since the network is trained using noiseless images, when noise is added to images for validation, it produces a larger area with random pixel values in the images with FoV of $60°$, compared to the images with FoV of $29°$, which may lead to poorer estimates. Also, it is observed that the type of concatenation does not influence the estimation result.

3.4.6 Regression layer with penalization

The regression layer was modified to penalize estimates in the cases where they fall outside the predefined range of the motion parameters. Each DoF can be penalized independently. The equation that describes the penalty for each DoF is

Input to the network	**RMSE of motion estimation**					
	Images without noise			***Images with noise***		
	Δ_x ***(mm)***	Δ_y ***(mm)***	Δ_θ ***(1°/100)***	Δ_x ***(mm)***	Δ_y ***(mm)***	Δ_θ ***(1°/100)***
Original Images - 29°/96 beams	2.75	1.71	5.3	5.92	4.16	12.0
Wide Images - 60°/512 beams - Horizontal concat	2.38	2.55	5.4	9.17	9.21	21.0
Wide Images - 60°/512 beams - Vertical concat	2.39	2.53	5.7	8.93	8.92	20.6

Table 3.5 Validation errors for 29° and 60° azimuth FoV images with different concatenation methods for PoseNet-Normx10.

$$\mathcal{P}_{j_k} = \begin{cases} \eta(\widehat{\Delta}_{j_k} - u_1)^2 & \widehat{\Delta}_{j_k} < u_1 \\ 0 & u_1 \leq \widehat{\Delta}_{j_k} \leq u_2 \\ \eta(\widehat{\Delta}_{j_k} - u_2)^2 & \widehat{\Delta}_{j_k} > u_2 \end{cases}, \tag{3.2}$$

where $j = x$, y or θ, η is a penalization parameter, and u_1 and u_2 are the boundaries defining the non-penalization interval. Then, the equation for the loss with penalization $\mathcal{L}_\mathcal{P}$ is

$$\mathcal{L}_\mathcal{P} = \frac{1}{2SW} \sum_{k=1}^{S} \left(\|\widehat{\Gamma}_k - \Gamma_k\|^2 + \mathcal{P}_{x_k} + \mathcal{P}_{y_k} + \mathcal{P}_{\theta_k} \right). \tag{3.3}$$

The network is trained using the rocky fields data set and the resulting RMSE for each DoF is shown in Table 3.6. The RMSE using penalization was obtained setting η to 100. Other values of η showed no significant difference in the estimation. u_1 and u_2 are set to -10 and 10, respectively, since the PoseNet-Normx10 architecture is used. Overall, there is a small difference between using and not using penalization for noiseless images but it is still better not to use the penalization. For noisy images, there is an improvement in translation estimation accuracy but the rotation estimation accuracy is worse. In general, introducing such a penalty does not provide significant improvement.

3.4.7 Transformation of polar to Cartesian coordinates

The images from the rocky field data set are transformed from polar to cartesian coordinates. The purpose of this transformation is to reduce the sideway RMSE assuming that the higher

Approach	RMSE of motion estimation					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
Without penalization	2.84	1.00	5.2	6.54	3.39	17.0
With penalization	3.02	1.09	5.1	5.80	2.67	26.6

Table 3.6 Validation errors for the PoseNet-Normx10 when trained with and without penalization.

error is due to the correlation with the rotation. The images in cartesian coordinates are used to train the PoseNet-Normx10.

Additionally, a network that combines images in cartesian and polar coordinates is designed. The combined network takes pairs of images in each coordinate system at the same time and trains a PoseNet-Normx10 network for each type of coordinates. The outputs of both subnetworks are concatenated to a fully connected layer before obtaining the final estimate. The validation results can be seen in Table 3.7. The RMSEs obtained using cartesian coordinates are slightly higher than the ones obtained with polar coordinates. This could be due to the interpolation error when transforming from polar to cartesian coordinates. However, the results for the combined network show some improvement in estimation of each parameter, especially for noisy images.

Approach	RMSE of motion estimation					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
Polar	2.84	1.00	5.2	6.54	3.39	17.0
Cartesian	3.36	1.08	5.1	9.75	3.33	11.1
Combined	2.60	1.07	4.7	4.99	2.27	8.6

Table 3.7 Validation errors for the PoseNet-Normx10 when trained with images in polar and cartesian coordinates and PoseNet-Normx10 with a combination of polar and cartesian coordinates.

3.4.8 Comparison with a deterministic method for attitude-trajectory estimation

The DL approach presented in this chapter is compared with the non-DL method presented in [79] which is described in subsection 1.2.3. In this method, the same three parameters of the sensor motion are estimated (Δ_x, Δ_y and Δ_θ), so it is possible to make a direct comparison with the DL approach.

For this comparison, only 100 out of the 8000 pairs of images from the validation data set have been randomly chosen from the rocky field test set. The full test set is not used due to the high computational time required by the non-DL approach to compute the estimates. The comparison is performed using images with and without noise. PoseNet-Normx10 and its version trained with noisy images are used for the comparison. Both methods are implemented in Matlab and run on the same PC with Intel Core i5-6500 processor and 8GB of RAM.

The results are presented in Table 3.8. They show a better performance of the DL techniques in almost all the parameters, except in the forward/backward motion estimation using images with noise.

Approach	**RMSE of motion estimation**					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
Non-DL	8.39	1.29	17.5	8.16	1.31	17.8
PoseNet-Normx10	3.33	1.08	5.4	7.03	3.25	16.5
PoseNet-Normx10wNoise	2.95	1.40	5.2	3.43	1.56	5.9

Table 3.8 Validation errors for the DL and non-DL techniques using 100 pairs of images.

The non-DL approach exhibits a robust performance regardless of whether the images are with or without noise. Overall, the best performance is achieved by the PoseNet-Normx10 trained with noisy images.

Furthermore, Table 3.9 shows the running time that each method requires to estimate from 100 pairs of images. It can be seen that the running time is significantly higher for the

non-DL technique compared to the DL network. The DL running time is 5.69 s which can be split in two steps: the image concatenation (1.27 s) and the estimation of the sonar motion (4.42 s).

Method	Running time (s)
Non-DL	2568.74
DL	5.69

Table 3.9 Running times of the non-DL and DL methods using a data set of 100 pairs of images.

Therefore, the DL method can obtain estimates for each pair of images every 56.9 ms. This suggests that this method could be implemented in real-time considering frame rates of an order of 20 fps (50 ms). However, a set of pretrained networks for different sonar parameters (such as the pitch angle and height) should be generated in advance.

3.4.9 Discussion about modifications of the DL networks

In this chapter, several approaches have been considered to train a network to estimate the motion of an underwater vehicle from acoustic images generated by a sonar. Five networks have been implemented and validated, then the best networks are modified to improve their performance. The following points summarize the results obtained:

- The five networks have been trained using quantized labels and images with and without noise. The best performance with noiseless images was provided by PoseNet and CNN-4PoseNet, whilst for images with noise, the best performance was provided by SfMNet.
- CNN-4PoseNet takes between 2 to 3 times longer to train, compared to the PoseNet and SfMNet.
- CNN-1b and CNN-4b show a poorer performance compared to the other networks.
- The analysis of the estimation errors shows a high correlation between the errors of the sideways motion and the rotation.

- SfMNet, PoseNet and CNN-4PoseNet show better results when trained using normalization of the labels rather than quantization. The best normalization is found to be within the range [-10, 10] with respect to the maximum motion values.

- The normalization results in a high improvement in the PoseNet performance when trained with noisy images and a similar performance to the quantization approach for noiseless images. The SfMNet shows similar results when trained with quantization and normalization of labels. The CNN-4PoseNet performance is observed to be worse with the normalization compared with the quantization. Therefore the PoseNet with normalization (PoseNet-Normx10) and the CNN-4PoseNet with quantization (CNN-4PoseNet-Qua) are selected for further modifications.

- A low level noise added to the images used for training the PoseNet-Normx10 and CNN-4PoseNet-Qua slightly increases the estimation errors when tested with noiseless images for both the networks. However, it considerably reduces the estimation error provided by PoseNet whereas the CNN-4PoseNet shows a similar performance when trained with noiseless and noisy images. Therefore PoseNet-Normx10 is selected for further optimization.

- Training the PoseNet-Normx10 with 3 concatenated images rather than 2 images, shows only slight improvement in the performance.

- A small dataset of wider images (higher aperture of the sonar in the azimuth dimension) was generated and used for training the PoseNet-Normx10. It shows estimation errors similar to the case of the original size images for noiseless images but an increase of the errors for noisy images. This can be due to larger image areas affected by noise but not carrying objects useful for motion estimation.

- Concatenating the images one besides the other (horizontally) or one above the other (vertically) is irrelevant for the network's performance. Both approaches show similar results.

- The loss function of the PoseNet-Normx10 was modified to penalize the cases when a parameter estimate falls outside the range [-10, 10]. The results show a similar performance when testing with noiseless images. With noisy images, it can be seen a small improvement for sideways and forward motion estimation but worse estimates for the rotation.

- The PoseNet-Normx10 trained with images in cartesian coordinates shows a slight increase in the estimation errors compared to the network trained with images in the original polar coordinates.

- The network that combines the images in polar and cartesian coordinates results in a smaller estimation error compared to the cases of using the polar or cartesian coordinates only.

- The PoseNet-Normx10 and PoseNet-Normx10wNoise have been compared with a non-DL method. Both DL networks achieve better estimation performance than the non-DL method.

- The computation time of the motion estimation is significantly reduced (by hundreds of times) when using the DL estimator instead of the non-DL method. The computational time for the DL approach is 56.9 ms for each pair of images, suggesting that this technique can be used in real-time applications.

Among all the networks and different approaches investigated in this chapter, the PoseNet-Normx10 proved to be one of the most reliable networks for motion estimation with sonar images. It showed a high estimation accuracy and high efficiency with noisy and noiseless images are used. Moreover, the PoseNet-Normx10 is one of the most feasible to implement, since it is has a low complexity architecture which helps to get low computation time. Therefore, the PoseNet-Normx10 is chosen to be used in trajectory estimation described in the following sections of this chapter.

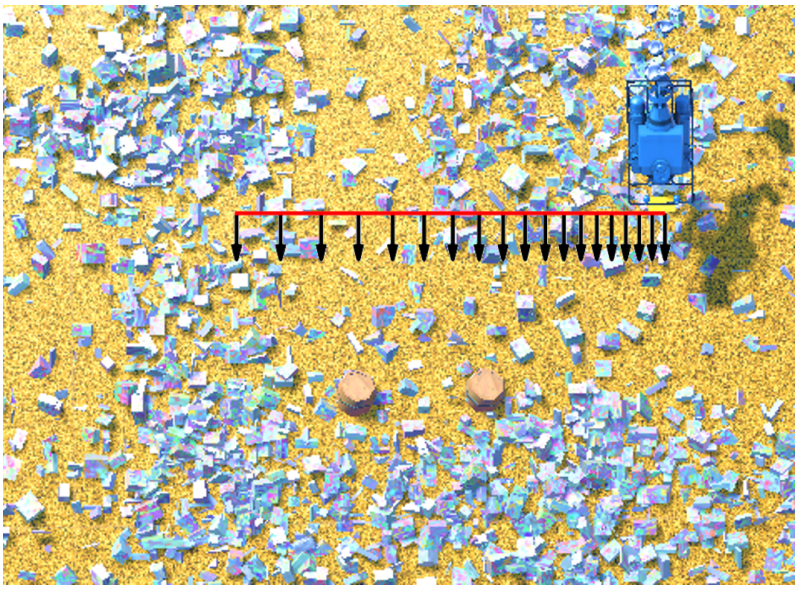


Fig. 3.6 Scenario for testing and generating images for mosaics. The red line represents the trajectory of the vehicle. The black arrows show for every 50 images the sonar orientation. The sonar keeps looking forward when moving along the trajectory.

3.5 Trajectory estimation

3.5.1 Trajectory estimation using synthetic data

The already trained PoseNet-Normx10 network is used for estimating the trajectory of a sonar vehicle. Using the sonar simulator, a new data set was generated for validating the trajectory estimator. The scenario shown in Fig. 3.6 was used and a trajectory with 904 noiseless images was created, keeping the same pitch angle and height from the seafloor. The motion of the sonar is in the sideways direction with a constant acceleration from 0.105 to 0.378 m/s along the trajectory. Every point of the trajectory is represented using global cartesian coordinates (x_i, y_i) and the vehicle orientation relative to the global scenario orientation (θ_i), which corresponds to 0° when facing in the y-axis direction; $i = 1, 2, 3, ..., T$, is the index for the image pairs and T is the number of images obtained on the trajectory. In total, $T - 1$ estimates are obtained for the trajectory. The first point of the trajectory is placed at the origin ($x_1 = 0$, $y_1 = 0$, $\theta_1 = 0$).

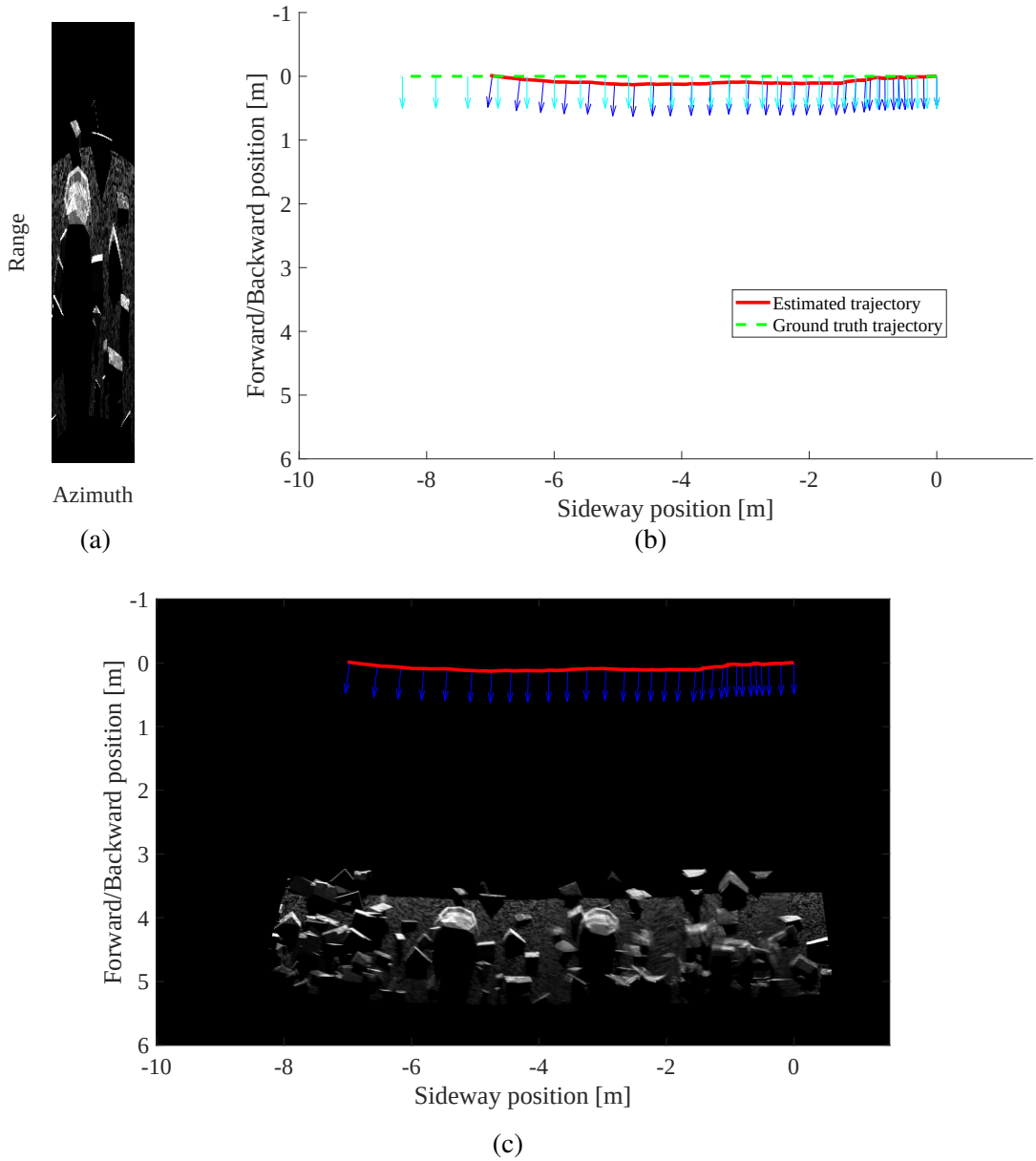


Fig. 3.7 (a) Single sonar image in polar coordinates used for constructing the mosaic. (b) Mosaic of a data set of 904 simulated images. (c) Green lines with cyan arrows represent ground truth trajectory and pose of the sonar, respectively, whilst the red line and dark blue arrows represent estimated trajectory and pose, respectively.

The coordinates are assumed to be on a plane parallel to the seabed. The trajectory points are calculated as follows [144]:

$$
\begin{aligned}
\theta_{i+1} &= \theta_i + \widehat{\Delta}_{\theta_i}, \\
x_{i+1} &= x_i + \widehat{\Delta}_{y_i} \sin(\theta_i + \widehat{\Delta}_{\theta_i}) + \widehat{\Delta}_{x_i} \cos(\theta_i + \widehat{\Delta}_{\theta_i}), \\
y_{i+1} &= y_i + \widehat{\Delta}_{y_i} \cos(\theta_i + \widehat{\Delta}_{\theta_i}) - \widehat{\Delta}_{x_i} \sin(\theta_i + \widehat{\Delta}_{\theta_i}),
\end{aligned} \tag{3.4}
$$

where $\widehat{\Delta}_{x_i}, \widehat{\Delta}_{y_i}$ and $\widehat{\Delta}_{\theta_i}$ are the x and y-translation and z-rotation estimates in the original units (metres and degrees) for each pair of images, respectively.

An example mosaic built from the images using the motion estimates can be seen in Fig. 3.7c. The images are mapped to the global coordinates according to the reconstructed trajectory. To produce the mosaic, the pixel intensities at each point are interpolated from the image pixels. The intensity of a pixel in an image is averaged over images when they overlap. For clarity, only the pixels that correspond to 24 center beams of each image are used for interpolation, except for the first and last images. In Fig. 3.7b, it can be seen that the shape of the trajectory and orientation are similar to the ground truth provided by the simulator. However, the estimates tend to be smaller than the ground truth.

Fig. 3.8a shows estimated and ground truth trajectories from a data set of 904 images where the motion is performed in the forward direction with a constant acceleration corresponding to the speed from 0.105 to 0.378 m/s. From the shape of the estimated trajectory, it can be seen that in the forward direction the estimates are more accurate. The size of the estimated trajectory in the forward direction is almost the same as the ground truth.

In Fig. 3.8b, a forward moving trajectory is displayed for the case of acceleration and deceleration. During the generation of this data set, the sonar vehicle moves always forward and accelerates and decelerates with a minimum speed of 0.063 m/s and a maximum speed of 0.357 m/s. The recovered trajectory is close to the ground truth.

3.5.2 Trajectory estimation using the ship's hull real data

The data set acquired by a real sonar (see subsection 3.2.3) is used to validate the performance of the DL approach for attitude and trajectory estimation. The PoseNet-Normx10 network is trained using the simulated ship's hull described in subsection 3.2.2. The network is trained

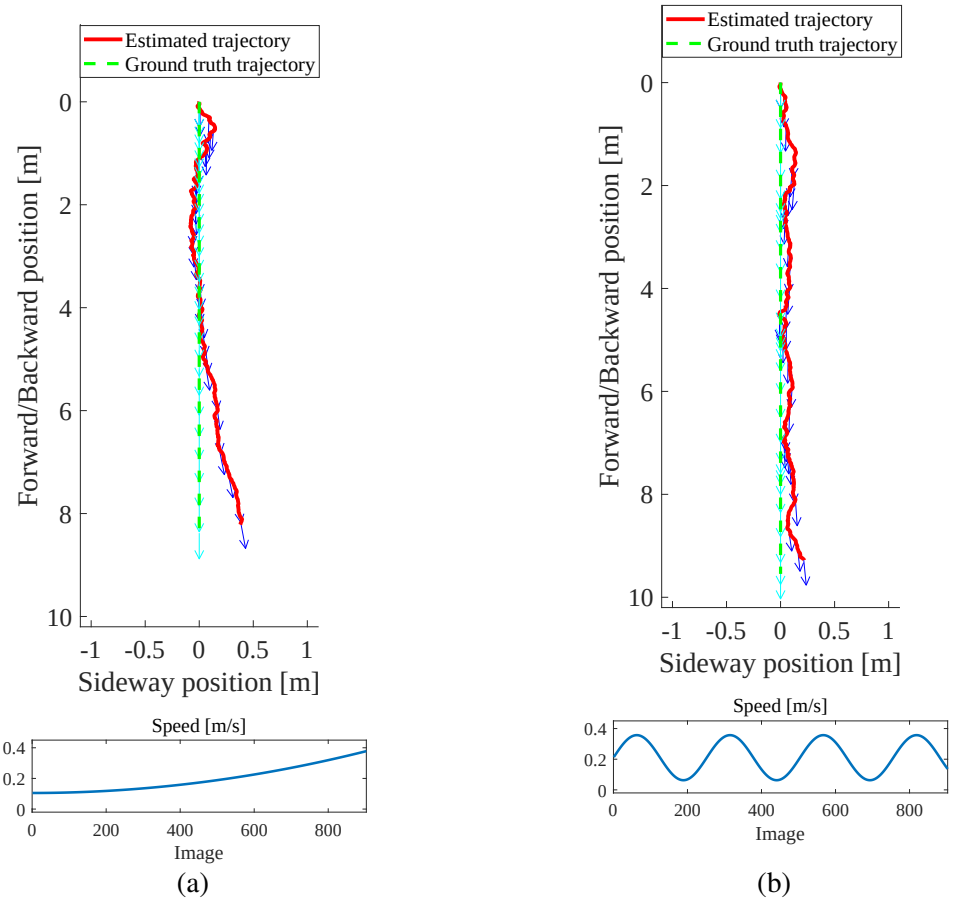


Fig. 3.8 (a) Trajectory of a sonar that moves in the forward direction with constant acceleration. (b) Trajectory of a sonar that moves in the forward direction with acceleration and deceleration.

three times applying different levels of noise on the images: noiseless, low level noise, which is the one used in subsection 3.4.3 and high level noise, which is the same as that used for validation of the networks, since this is the level measured in the same real data set in [118]. A comparison of training with the three different levels of noise is presented in Table 3.10. The network trained with the high level of noise is selected for the trajectory estimation since it shows a better RMSE in the validation with noise, which is assumed to be similar to the real data set.

The already trained network is fed with the real data set to estimate the displacements between each pair of consecutive images. The sonar trajectory is recovered from the estimates as described in subsection 3.5.1. Based on the shape and length of the obtained trajectory, the

Approach	RMSE of motion estimation					
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(1°/100)*
	Images without noise			***Images with noise***		
Noiseless images	2.99	1.63	4.5	4.41	2.72	6.7
Low level noise images	2.93	1.83	4.5	3.98	2.66	5.9
High level noise images	3.20	1.75	5.2	3.22	1.77	5.2

Table 3.10 Performance of the PoseNet-Normx10 network for the simulated ship's hull data set with different levels of noise

estimated trajectory seems to be shorter than expected in the sonar sideways direction. This assumption is based in the works [29, 79], which use the same real data set to estimate the trajectory of the sonar. The length of the ship's hull is estimated to be approximately 10 m. Therefore, all the estimates in the sideways direction are multiplied by a constant coefficient 2.3 to provide a better match to the hull size. The need of such coefficient could be related to inaccuracies of setting the sonar parameters that are used by the simulator to create the training data set, like the height and sonar pitch angle.

By using the compensated trajectory, the images are merged into the mosaic shown in Fig. 3.9b. One single sonar image that is used to build the mosaic is presented in polar coordinates in Fig. 3.9a, corresponding to a fragment of the ship's keel, at the center-right of the mosaic. From the mosaic, it can be seen that motion estimates are reasonably accurate, with a few distortions in the image. Some sacrificial anodes and the keel are clearly recognizable.

3.5.3 Trajectory estimation using the dam inspection real data

The data set described in subsection 3.2.4 was acquired by a real sonar during a dam inspection. The already trained PoseNet-Normx10wNoise is fed with the images from the data set to estimate the trajectory followed by the sonar. This network is selected due to similarity of objects in the images of the real data set with the rocky fields training set. However, the real data set presents significant differences that affect the estimate, such

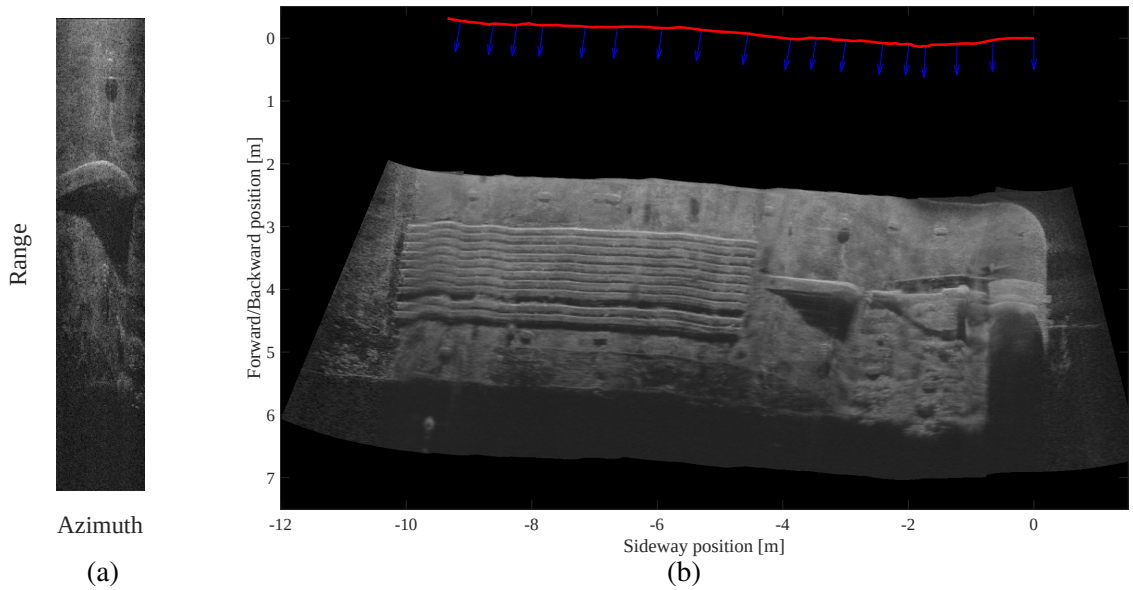


Fig. 3.9 (a) Single real sonar image in polar coordinates used for constructing the mosaic. (b) Trajectory and mosaic obtained using 520 images of the ship's hull real data. The network used for estimation is the PoseNet-Normx10 trained with the simulated ship's hull data set with noise.

as the pitch angle of the sonar and its distance from the seabed. Therefore, the estimates obtained by the network are scale compensated to generate a more accurate trajectory. The compensation multiplies the estimates in the forward/backward motion and the sideways motion by a constant coefficient (4.0 and 9.0, respectively). The size of the sonar images is 1344×128 pixels and the FoV of the sonar is 30° along 128 beams in the azimuth dimension. To match the input size of the network, only pixels in a window of the size 512×96 in the center of each image are used for estimation. The estimates are used to generate the full sonar trajectory. Then the sonar images are merged into a mosaic (Fig. 3.10) by following the estimated trajectory. The dam wall can be seen as a continuous white line at the bottom of the mosaic. The data set contains sharp discontinuities, also the presence of fish in some images can cause low quality of the estimates and affect the full trajectory estimation. However, it can be seen that using a trained network, it is possible to produce a decent mosaic even if the sonar features of the training set are different. The mosaic is highly similar to the mosaic

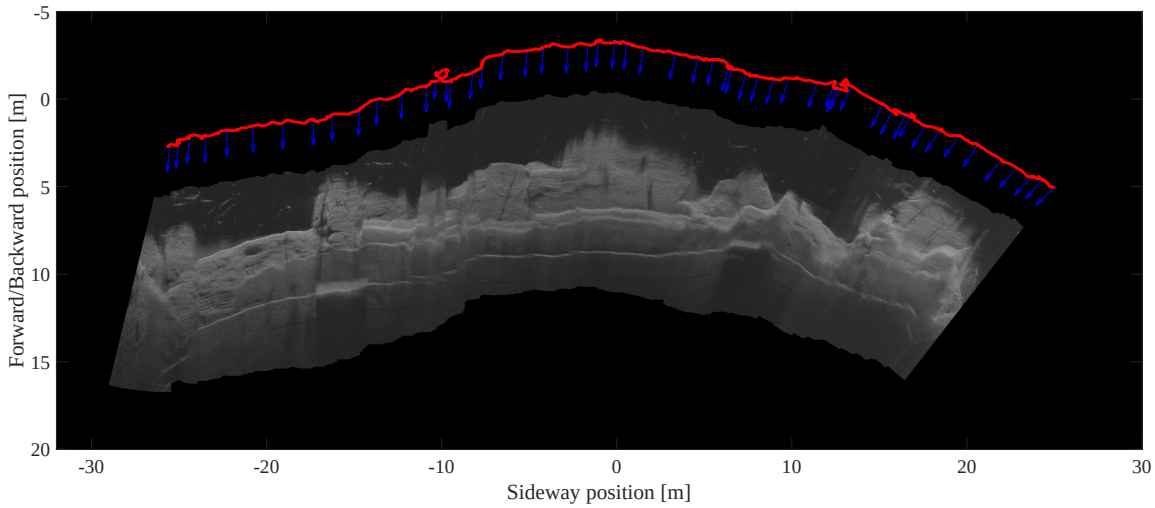


Fig. 3.10 Trajectory and mosaic obtained using 1596 images of the dam inspection data set [3, 4]. The network used for estimation is PoseNet-Normx10wNoise trained with the simulated rocky field data set. The sonar sensor trajectory is shown in red and the attitude as a blue arrow every 30 images.

built in [145], which uses the deterministic method for the motion estimation followed by regularized P-splines for smoothing the trajectory.

3.6 Motion estimation using images from two sonar sensors

The DL-based motion estimation method presented in this chapter significantly reduces the complexity and processing time compared to the deterministic method in [79]. However, higher estimation accuracy is required for some applications, e.g., synthetic aperture sonars [146–148]. Furthermore, the sonar configurations presented so far are limited to the use of FLS images. This means that the motion estimates rely on the information acquired by a single sonar receiver. Therefore, the aim in this section is to investigate the use of two sensors separated from the sonar transmitter to find out how it can improve the accuracy even further. For this, the sonar simulator is modified to allow acoustic images to be generated for more complicated sonar configurations and then these images are used for training and validation of DL networks.

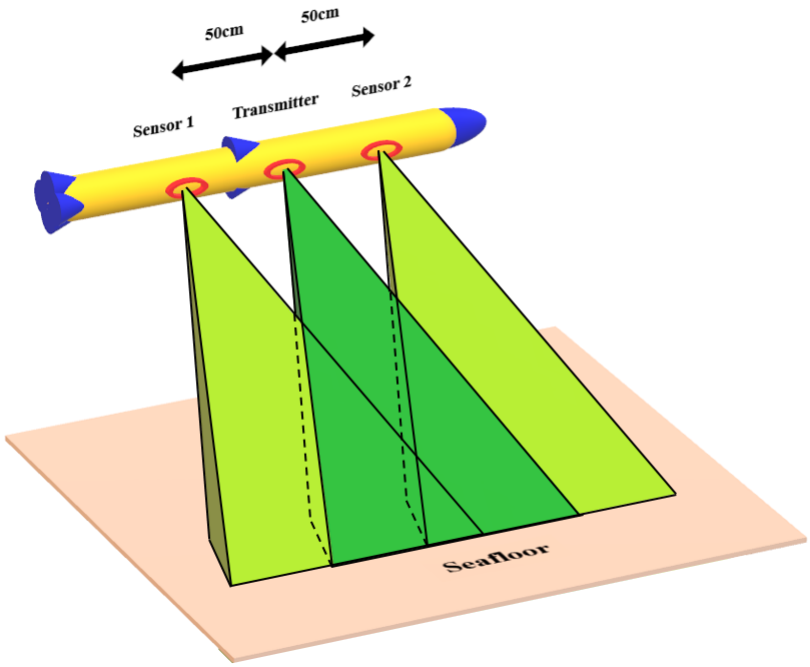
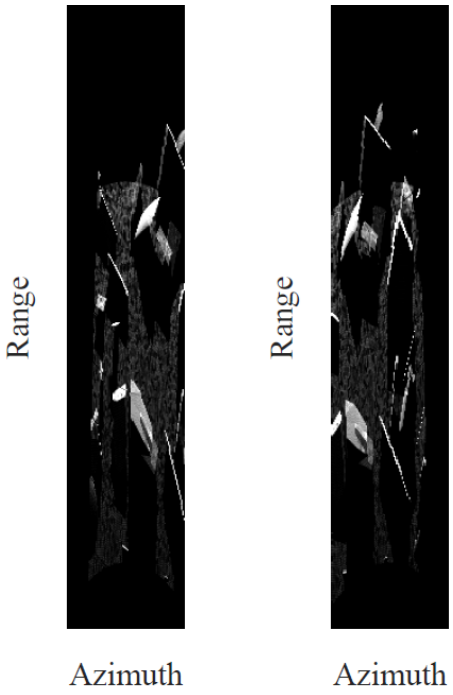


Fig. 3.11 The transmitter and two sensors facing perpendicularly to the direction of the underwater vehicle.

3.6.1 Data set generation

The simulator from Chapter 2 only uses a single sonar sensor with the same transmit-receive antenna. The sonar simulator was expanded to separate the sonar transmitter from the receiver. Also, the capability to simulate a sonar with multiple sensors in different positions at the same time is added. A transmitter illuminates the environment while the sensors generate the sonar images. This is shown as the dark green beam in Fig. 3.11. An image is generated from each of the sensors. Examples of sonar images generated by the sensors are shown in Fig. 3.12. Since they are situated on each side of the transmitter, they have a slightly different point of view of the scenario. In the images, it can be seen that some area of the image is totally dark. For sensor 1, this area appears on the left and for sensor 2, the area appears on the right. This is because the FoV of the transmitter does not totally overlap with the FoV of the sensors, so this portion of the sensor's FoV does not receive signals from the transmitter.



(a) (b)

Fig. 3.12 Examples of sonar images generated by the two sensors at the same time. (a) Image generated by sensor 1. (b) Image generated by sensor 2.

For the experiments in this section, the displacement of the sensors between consecutive images is the same that was previously used in this chapter, which is described by a vector $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]$, representing translations along the x and y-axes and rotation around the z-axis (denoted by θ), respectively (see Fig. 3.3a).

To generate data sets, the vehicle moves through many positions by following a randomly generated trajectory. The simulated underwater environment is the rocky field described in subsection 3.2.1. Pairs of images generated in consecutive positions of the trajectory are concatenated into a single image. Each concatenated image is associated with a displacement label to make a training sample. The label corresponds to the vector Δ. The three elements in the labels are normalized to the range from -10 to 10 with respect to their maximum values, as described in subsection 3.4.2. A data set of 20,000 pairs of concatenated images is generated for each sonar sensor.

3.6.2 Experiments using one and two sensors

Sonar configurations

With the modified simulator, four sonar configurations are built:

1. **One Sensor (same place):** This configuration is the same that was previously used in this chapter. One transmitter with a single sonar sensor is used with no separation between them.

2. **Two sensors:** One transmitter (dark green beam in Fig. 3.11) and two sonar sensors (two light green beams in Fig. 3.11) are placed on an underwater vehicle to side look perpendicularly to the forward motion of the vehicle as shown in Fig. 3.11. The distance between the transmitter and each sensor is 50 cm.

3. **Sensor 1:** Only the images generated by the sensor 1 from the two sensors configuration are used. The idea of using one sensor is to investigate how it can impact in the estimation accuracy.

4. **Sensor 2:** Only the images generated by the sensor 2 from the two sensors configuration are used.

For all the configurations, the FoV of the transmitter and the sensors is $29^\circ \times 14^\circ$ (azimuth and elevation angles, respectively), with 96 beams in the azimuth and a pitch angle of 35°. This is based on the parameters of the DIDSON 300 sonar [2].

For the experiments in this section, the maximum displacement between consecutive two images is 2.0 cm and 0.45° for the translations and rotation, respectively. The height of the sensors from the seafloor is 2.5 m. The sonar image size is 512×96 pixels.

DL network and training parameters

Given that the PoseNet has been found to be well suited motion estimation after optimizing the network parameters to get best possible performance, the same network is used to validate the motion estimation using the new proposed sonar configuration.

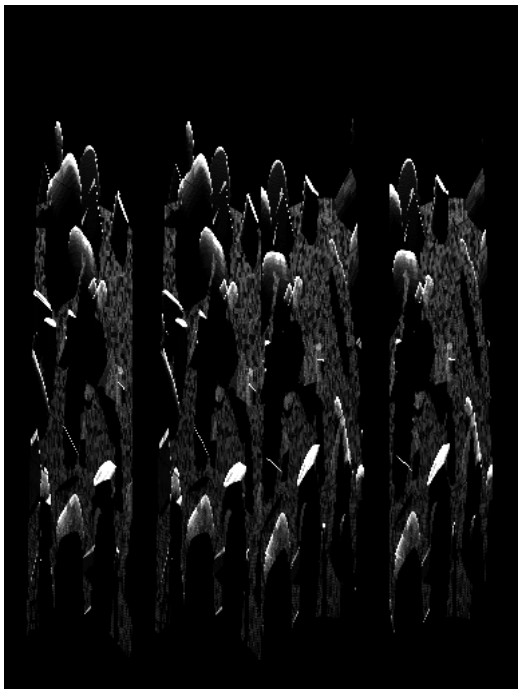


Fig. 3.13 Concatenated image using four sonar images. The left half comprises the two images generated by sensor 1 and the right half comprises the two images generated by sensor 2.

For the two sensors configuration, the already concatenated images from each sensor are concatenated with the corresponding concatenated images of the other sensor to make a larger image of four concatenated images as shown in Fig. 3.13. This larger image is put into the network. For the one-sensor case, the pairs of concatenated images are directly put into the DL network. The data sets for both the cases are split into 95% and 5% for training set and validation set, respectively. The DL networks are trained in MATLAB. The Adam optimization algorithm [143] is used for training. The learning rate starts at 0.0001 and halves every 12 epochs until the validation loss converges. During the training, a dropout regularization with a rate of 50% is applied.

The sonar images generated by the simulator are noiseless. Two approaches are followed for training the networks. One consists in training with the noiseless images and the other consists in training with the same images, but with a low-level of noise added to their pixels. The noise is generated as follows: (i) the pixels of acoustic shadows in the images are

modified with additive Gaussian noise with a mean and standard deviation of 4% and 2% of the maximum pixel value, respectively. (ii) The remainder of the pixels are affected by adding noise with the Rayleigh distribution with a scale parameter of 4% of the maximum pixel value, thus representing the scattering noise.

After the networks have been trained, the estimation accuracy is validated using either the noiseless images or images with a high-level of noise based on measures of noise in real DIDSON sonar images described in [118]. The high-level noise has a Gaussian distribution with the mean and standard deviation of 13.72% and 3.14% of the maximum pixel value, respectively, and a Rayleigh distribution with a scale parameter of 13.72% of the maximum pixel value.

3.6.3 Results and discussion

Table 3.11 presents results of training the networks with noiseless and low-level noise images in the terms of the RMSE obtained when validating with noiseless and high-level noise images.

Training approach	**RMSE of motion estimation**					
	Validation on noiseless images			***Validation on high-level noise images***		
	Δ_x ***(mm)***	Δ_y ***(mm)***	Δ_θ ***(1°/100)***	Δ_x ***(mm)***	Δ_y ***(mm)***	Δ_θ ***(1°/100)***
With noiseless images						
One sensor (same place)	2.69	2.74	5.4	5.73	5.82	11.8
Two sensors	2.17	1.42	4.4	7.47	6.13	15.6
Sensor 1	2.80	1.99	6.2	9.68	8.52	19.0
Sensor 2	3.15	2.40	7.2	9.89	7.94	22.2
With low-level noise images						
One sensor (same place)	3.52	3.38	7.6	5.75	5.80	12.5
Two sensors	3.05	2.33	7.0	4.37	3.34	8.9
Sensor 1	5.14	4.08	11.3	5.18	3.91	11.5
Sensor 2	4.43	4.23	9.9	5.76	5.64	13.1

Table 3.11 Validation RMSE when training the DL network with noiseless images and with low-level noise images.

It can be seen that better performance is obtained by the network trained with two sensors over the networks trained with one sensor. For the y-axis in the training and validation with noiseless images, the RMSE for the one sensor (same place) configuration is 2.74 mm and for the two sensors configuration is 1.42 mm, which is a reduction of almost twice. The other parameter estimates are also improved when training with the two sensors configuration.

The only case when the two sensors configuration presents a higher RMSE compared to the one sensor (same place) configuration is when training with noiseless images and validation with high-level noise images. This can be caused by the black areas on the side of each sonar image which do not provide information about the motion. It is possible that since this area is always black (0 value), the network learns to ignore that part of the images for the motion estimation, but when randomly generated noise is added, it affects the estimates. This issue is eliminated when training with noisy images, even if it is not the same level of noise that is used for validation. The same problem seems to occur when using the networks from sensors 1 and 2. The images from these sensors have a black area that affects the estimation accuracy.

For all the configurations, the best estimates are for the y-axis. This can be due to the high correlation between estimates of translation along x-axis and the rotation described in subsection 3.4.1, which affects the estimation accuracy. Therefore estimates along x-axis and the rotation are less accurate than the estimates for the y-axis.

Training with low-level noise images reduces the RMSE of the two sensors configuration while the RMSE of the configurations of one sensor is not reduced. This suggests that using the two sensors configuration is more suitable for motion estimation with real data, since the noise level in this case is the same as measured from real sonar images.

When training, the DL network parameters are tuned to provide the best performance for the one sensor (same place) case; the same network with the same parameters as used in the previous sections of this chapter. However, a better performance is obtained by the network trained with two sensors even without optimizing the parameters.

3.7 Conclusions

An attempt to use DL approaches for trajectory estimation using sonar images is presented. The basis of the method is the use of large volumes of synthetic images generated by a sonar simulator to train the DL networks, and then the application of the already trained network to real data. The use of synthetic data provides the ground truth data needed to perform supervised learning and quantitatively validate the motion estimates. Several DL architectures and their modified versions are presented, implemented and compared. The results obtained using different network architectures show that a millimetre accuracy for translation motion and below 0.1° for rotation motion can be achieved. Networks trained with simulated data and then used for estimation with real data sets can obtain reasonably good estimates even when the sonar features between the training set and the real data sets are different, by taking into account scale compensation parameters. The PoseNet architecture and its variations present the best results compared to the other networks. The PoseNet with normalized labels is applied to real sonar data sets, obtaining a good trajectory estimation that is used to construct mosaics by merging the images. The quality and realism of the simulated images used for training are important in the motion estimation applied to real data. For example, the use of the appropriate level of noise in the synthetic training images improved the accuracy of the trajectory estimation. The DL network obtained more accurate motion estimates using significantly lower computation time compared to the deterministic algorithm. However, the DL network requires retraining with a new data set if the sonar parameters change. One solution for this can be the use of multiple pretrained networks whose training data set is adjusted to different sonar parameters.

A technique that combines sonar images from two sensors rather than using images from only one sensor is presented as an effort to improve the motion estimation accuracy obtained with a single sensor. The two sensors configuration shows an improvement in the estimation accuracy compared to the one sensor configuration, even without tuning the DL training parameters to try to optimize the estimation. For instance, there is an RMSE reduction of

almost twice for the y-axis movement, while the RMSE for the other types of movement is also reduced.

In this chapter, it has been shown that the complexity of the DL network can be made as low as, in some situation, the real-time processing is feasible. However, to achieve a higher localisation accuracy, more complicated processing might be useful. Therefore, it is important to develop more sophisticated techniques, which would be both more accurate and, at the same time, possess less complexity. A technique that provides a higher localisation accuracy and has a reduced computational load is proposed in the next chapter.

Chapter 4

Sonar FoV segmentation for motion estimation using DL networks

The DL techniques for motion estimation proposed in Chapter 3 have shown promising results for underwater navigation based on analyzing sonar images. However, real-time applications are still a challenge. In this chapter, low-complexity techniques for motion estimation based on the use of images obtained by a sonar looking down to the seafloor are investigated. Various configurations of the sonar beams are considered according to portions of the FoV covered and the images from those portions are converted into vectors to reduce their size before applying them to a DL network. To investigate the new configurations, the sonar simulator described in Chapter 2 is modified to allow images to be generated for more complicated sonar configurations and to train and validate the DL networks. In Section 4.1, an introduction to the preprocessing method described in this chapter is presented. In Section 4.2, an adaptation of the PoseNet from Chapter 3 and description of the data sets to train the network are presented. In Section 4.3, the proposed motion estimation methods with different portions of the sonar FoV are described. Section 4.4 presents results and discussion for the DL network validation and trajectory estimation with simulated and real data. Section 4.5 presents a method for estimating the height from the seafloor of vehicle using a DL network. Finally, conclusions are given in Section 4.6.

The work in this chapter is presented in the paper: J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, "Sonar FoV segmentation for motion estimation using DL networks," *IEEE Access*, vol. 10, pp. 25 591–25 604, 2022.

4.1 Introduction

As described in Section 1.2, the combination of multiple traditional methods for underwater navigation can alleviate the problems associated with the methods when operating alone but at the cost of increasing the whole system complexity. In this chapter, a motion estimation method that can be suitable for real-time applications, keeping a high accuracy is investigated. It is similar to the DL method from Chapter 3, but instead of using a FLS, the source of information is a looking down sonar whose measurements are preprocessed in different ways. This simplifies the DL network reducing the size of data that the DL network processes and it is done in two steps: (i) a portion of the sonar FoV is selected and used to generate images; (ii) the images are converted into vectors (reduced size) and grouped together as part of the preprocessing.

4.2 DL network and sonar simulator

This section presents the DL network for motion estimation as well as the description of the training data sets and the simulated underwater environment.

4.2.1 DL network architecture using sonar images as the input

In Chapter 3, the PoseNet network [120] was found to be the most efficient for estimating the motion of an underwater vehicle using FLS images. The parameters of the network such as the number of convolutional layers, the size of kernels, etc, have been optimized for this task. The resulting optimized PoseNet for motion estimation with sonar images was named PoseNet-Normx10. The architecture of the PoseNet-Normx10 can be seen in Fig. 3.4b (for simplicity, in this chapter it will be called PoseNet). Parameters of the layers can be seen in

Table 4.1. The values in the column "Output size per channel" depend on the input image size. The values in the table are shown for the image size of 512×2000 pixels (two concatenated images of 512×1000 pixels), which is the size of the first configuration described below in Section 4.3. An output regression layer is connected to the average layer to generate motion estimates for the 3 DoF, since 3 DoFs are still considered in the experiments in this chapter: forward/backward motion, sideways motion and yaw rotation, denoted by $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]$, respectively. It is considered that the vehicle maintains a constant height from the seafloor when it moves and roll and pitch rotations are too small to be considered [29, 79].

Layer	Kernel	Channels	Stride	Output size per channel	ReLU & Batch norm
Conv1	7×7	32	2×2	256×1000	Yes
Conv2	5×5	64	2×2	128×500	Yes
Conv3	3×3	128	2×2	64×250	Yes
Conv4	3×3	256	2×2	32×125	Yes
Conv5	3×3	512	2×2	16×63	Yes
Conv6	3×3	512	2×2	8×32	Yes
Conv7	3×3	512	2×2	4×16	Yes
Conv8	3×3	1024	2×2	2×8	Yes
Conv9	3×3	1024	2×2	1×4	No
Average	4×4	1024	1×1	1×4	No

Table 4.1 Parameters of PoseNet layers as presented in Chapter 3.

4.2.2 Generation of data sets from the simulator for training the DL network

The sonar simulator described in Chapter 2 uses ray-tracing to generate sonar images. A simulated sonar sends multiple rays in a predefined FoV specified by aperture and elevation angles, and the maximum range (see Fig. 4.1a). The rays are equally separated in the elevation and aperture angles. The ray separation is a parameter that can be set in the simulator and it is independent for the aperture and elevation dimensions (see Fig. 4.1b).

The simulator is used to generate large volumes of data with ground truth information for the position and orientation of the sonar vehicle. Simulated data is used to train a DL

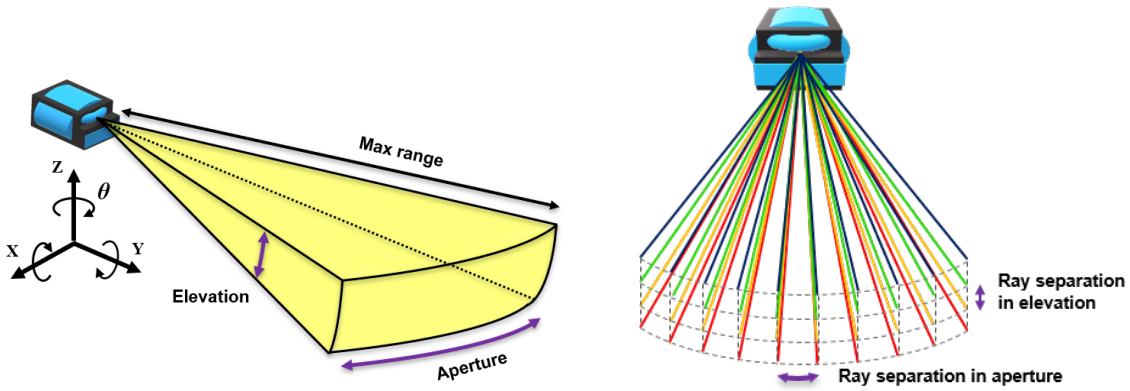
network effectively, since such training requires a large amount of ground truth data, which would be almost impossible to collect in real sea experiments. In this chapter, new sonar FoV configurations are considered and implemented in the simulator.

The training data sets are generated by concatenating pairs of consecutive sonar (Q=2) images into a single image. The images obtained from the simulator are represented in polar coordinates, they are not transformed to cartesian coordinates. The PoseNet described in subsection 4.2.1 is trained using supervised learning. Therefore, labels with information on the vehicle displacement need to be associated with the training samples.

As in Chapter 3, the simulated underwater vehicle moves in a random scenario to generate a data set. For all the experiments in this chapter, the maximum displacement between two positions (corresponding to two consecutive sonar images, and assuming the maximum vehicle velocity 0.42 m/s) is $\pm$ 20 mm for translation in one direction and the maximum rotation is $\pm$ 0.45°. Each scenario is generated as described in subsection 3.2.1. A scenario consists of a flat surface representing the seabed, covered with randomly placed rocks. An example of a simulated scenario is shown in Fig. 4.2. Given that the trajectory is also randomly generated, it is unlikely that the vehicle passes through the same position and orientation more than once. However, to avoid any possible similarities, several different scenarios are used. Each data set is generated using 17 scenarios. The data sets are shuffled and split into 95% and 5% for the training and validation, respectively.

4.3 Processing whole sonar images and images compressed into vectors

A sonar which looks down with a FoV of 100°$\times$100° is simulated. The height (distance from the sonar to the seafloor) is 2.5 m as shown in Fig. 4.3. It can be seen that the vehicle forward direction corresponds to the x-axis, the sideways direction to the y-axis and the upward direction to the z-axis, which is the rotation axis associated with θ. This coordinate system will be used in the rest of the experiments in this chapter to analyze and validate the proposed techniques.



(a) (b)

Fig. 4.1 (a) The sonar in the simulator is characterized by the aperture and elevation angles, and the maximum range that it can measure. (b) The simulated sonar FoV is made of multiple rays that are sent from the sonar with a defined separation in the aperture and elevation dimensions.

Fig. 4.2 Example of the simulated underwater environment with a moving vehicle.

Two techniques for motion estimation are considered:

1. **Processing the whole image (intensity over the aperture-range plane):** Images are generated by compressing the FoV in elevation. This is done by adding up all reflections from objects which are at the same aperture angle and range from the sonar. This process eliminates the elevation dimension.

2. **Processing a compressed image, i.e., a vector (intensity over the range):** In addition to the FoV compression in the elevation dimension, the image is also compressed in the aperture dimension. The result of this compression is a vector with the summation of all the reflections from objects that are located at the same distance from the sonar. The aim of this technique is to reduce the amount of data representing the sonar image and thus reduce the complexity and computation times of the motion estimation system.

4.3.1 DL network architecture using vectors as the input

When using the vector technique, every sonar image is converted into a vector, therefore, the input of the DL network is the concatenation of two vectors (one vector obtained from each image), rather than two images. The kernel and stride of each convolutional layer is now changed to avoid a reduction of the number of columns, where each column corresponds to a vector. The summary of the DL network is shown in Table 4.2. This modified version of the PoseNet is called *PoseNetVec*.

The next subsections describe the use of different portions of the sonar FoV to find a configuration that gives the best estimates of the vehicle motion.

4.3.2 Compression of the FoV into a single vector

The $100^\circ \times 100^\circ$ sonar FoV described above is generated by multiple rays that are sent by the simulated transducer. The separation between rays for the transducer in the simulation is 0.1° in each direction. The value of 0.1° was observed to give a moderate computation time with a good resolution.

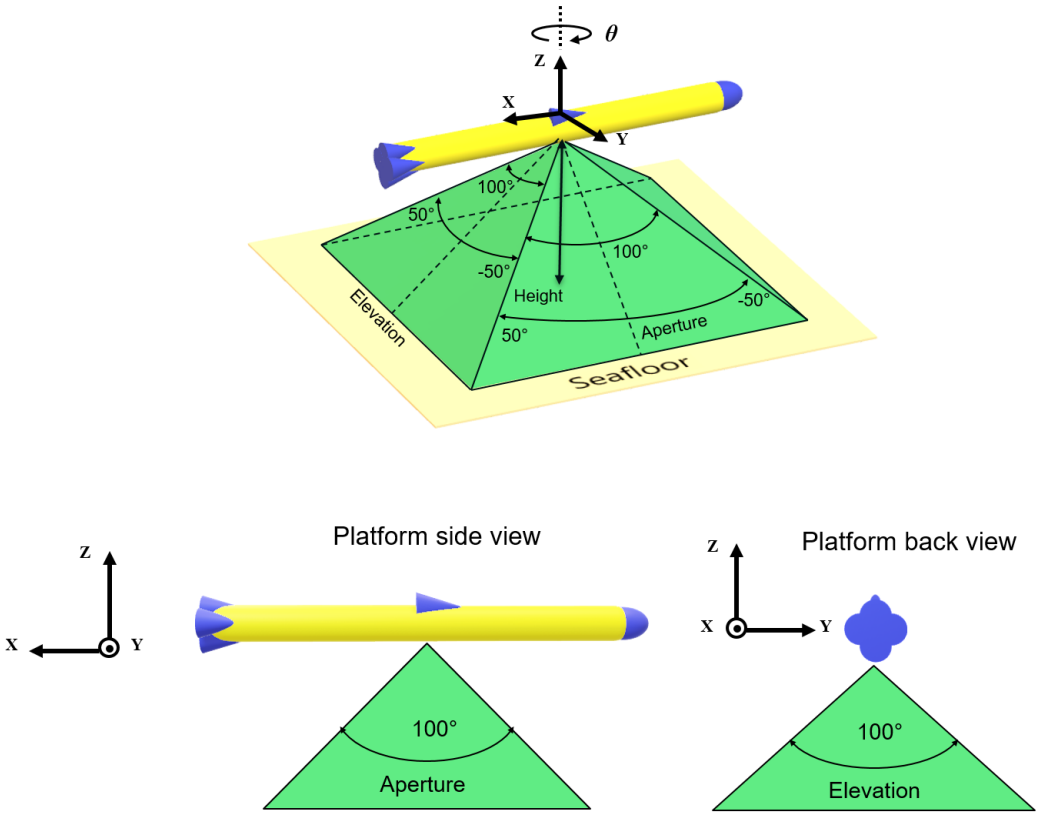


Fig. 4.3 Sonar with a looking down FoV of 100°×100°. The sonar is mounted on a vehicle whose forward direction corresponds to the negative direction of the x-axis.

Layer	Kernel	Channels	Stride	Output size per channel	ReLU & Batch norm
Conv1	7×2	32	2×1	256×2	Yes
Conv2	5×2	64	2×1	128×2	Yes
Conv3	3×2	128	2×1	64×2	Yes
Conv4	3×2	256	2×1	32×2	Yes
Conv5	3×2	512	2×1	16×2	Yes
Conv6	3×2	512	2×1	8×2	Yes
Conv7	3×2	512	2×1	4×2	Yes
Conv8	3×2	1024	2×1	2×2	Yes
Conv9	3×2	1024	2×1	1×2	No
Average	4×4	1024	1×1	1×2	No

Table 4.2 Parameters of PoseNetVec layers for the compressed images.

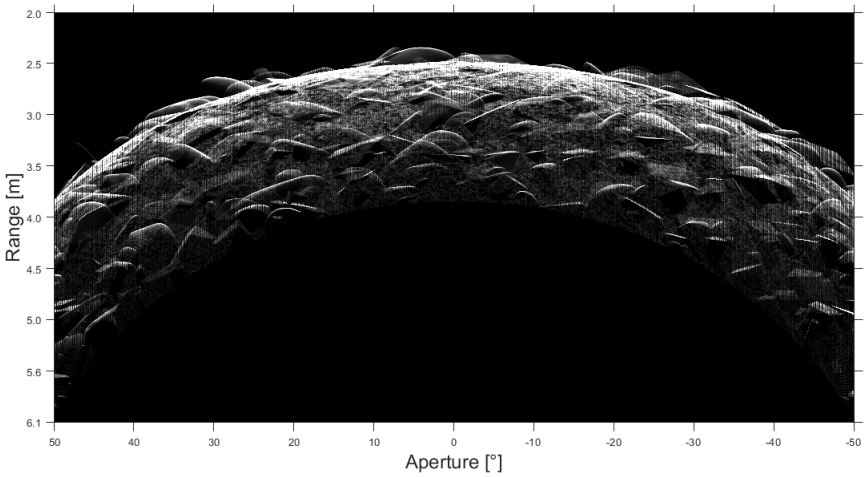


Fig. 4.4 Image generated using the FoV of $100^\circ \times 100^\circ$ and a ray separation of 0.1°. This image is compressed in elevation.

To produce the images, the range dimension is discretized into 512 values while the aperture dimension is discretized into 1000 values. The size of the resulting image is 512×1000 pixels (range by aperture). An example of a generated image can be seen in Fig. 4.4. The pixels in the first row of the image correspond to the range of 2 m from the sonar and the pixels in the last row correspond to the range 6.1 m. These values are selected because out of side these boundaries there is no information collected by the sonar. For the vector technique, a vector of size 512 is computed by adding up all the pixel values on each row of the image. This produces a vector of 512 range levels, where the first and last levels correspond to ranges (distance from the sonar) of 2 m and 6.1 m, respectively.

4.3.3 Compression of half of the FoV into a vector

Given that the sonar is looking down and that images are generated by adding up the data in the elevation angle, the resulting images have objects from both sides of the vehicle, by overlapping the objects that are at the same distance and aperture angle. An example of overlapping can be seen in the third image in Fig. 4.5. If the rays of only one single side of the vehicle are used to generate the image, the overlapping can be avoided. For example, the

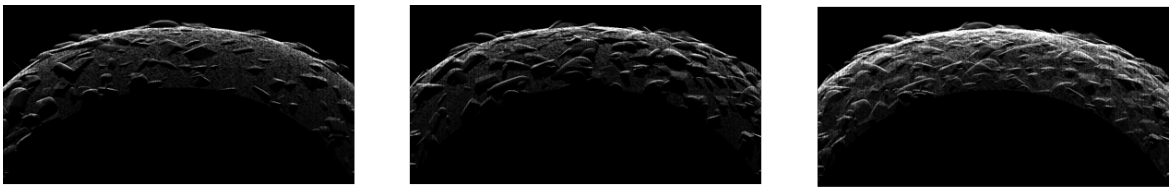


Fig. 4.5 Images generated using elevation angles (from left to right) [-50, 0], [0, 50] and [-50, 50]. In the third image, which is the sum of the first two images, it can be seen the overlapping of objects from the first two images.

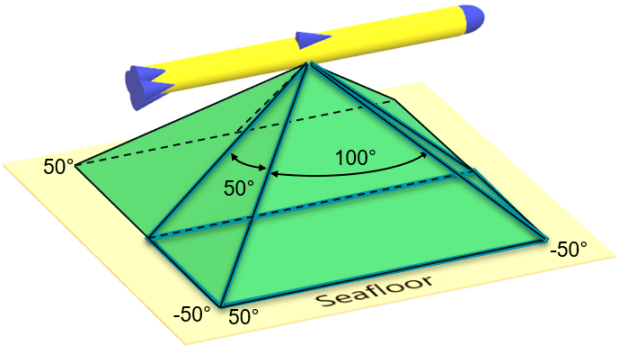


Fig. 4.6 The FoV for one side of the underwater vehicle. The size of the FoV is $100° \times 50°$.

first and second images in Fig. 4.5 are generated with the rays of the right and left side of the vehicle, respectively. If both images are combined, the result is the third image.

The overlapping causes the following: when the vehicle rotates, half of the objects in the images are displaced to one side of the image and the other half of the objects are displaced to the other side of the image, which introduces an ambiguity in the estimation process.

To avoid the overlapping, images are generated using the rays of only one side of the vehicle. Fig. 4.6 highlights with a blue line the corresponding portion of the FoV. The size of the generated images and vectors are the same as in the case of using the whole FoV.

4.3.4 Compression of four FoV quadrants into four vectors

Rather than separating the sonar FoV into two segments, a new configuration that splits the whole FoV into four quadrants is now used. Fig. 4.7 shows the quadrant segmentation.

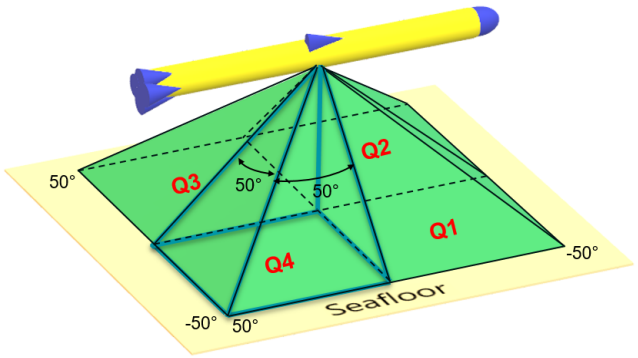


Fig. 4.7 Segmentation of the FoV into 4 quadrants. Each quadrant points to a different direction and has a portion of the FoV of $50° \times 50°$.

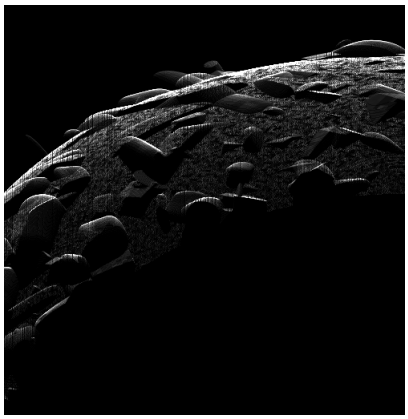


Fig. 4.8 Image generated using the information from segment Q4.

The images are still generated by adding up the information in the elevation angle corresponding to each quadrant. An example of an image generated using the quadrant segmentation can be seen in Fig. 4.8, which corresponds to Q4.

With the quadrant configuration, two variants are considered to train the DL network:

- **Single quadrant:** The PoseNet is trained using images from quadrant Q4.
- **Four quadrants:** The PoseNet is trained using images from all 4 quadrants. This variant is equivalent to 4 sonars pointing to different directions. For motion estimation, 8 images (2 from each quadrant) are concatenated into a single image which is the input to the DL network. An image example with the concatenation of 8 images and its

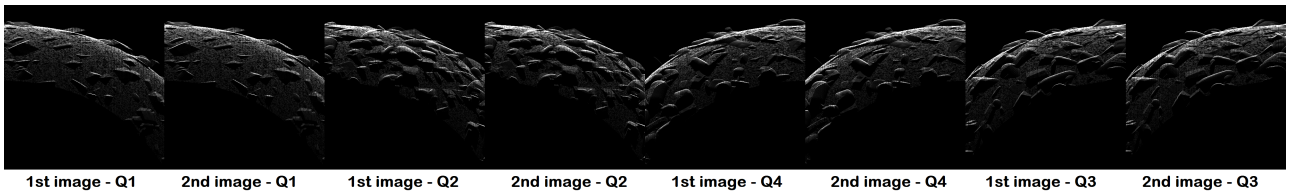


Fig. 4.9 Concatenation of 8 images (2 consecutive images from each quadrant) as input to the DL network.

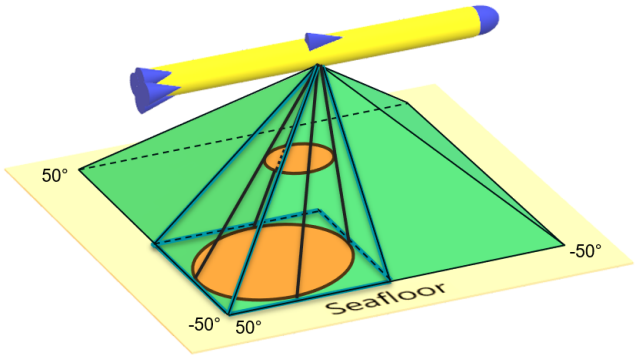


Fig. 4.10 Representation of the circular segment.

order is shown in Fig. 4.9. For the vector technique, the DL network receives as input, 8 concatenated vectors (2 from each quadrant), which is equivalent to a 512×8 matrix.

4.3.5 Compression of circular segments of the FoV quadrants into vectors

The four segments defined in subsection 4.3.4 are now cut to a circular shape to form a cone as shown in Fig. 4.10. The rays within the cone are used to generate the sonar images. An example of an image obtained with the circular segment (Q4) is shown in Fig. 4.11. The image is very similar to the one for a quadrant in Fig. 4.8, with the difference that there is slightly less data, as seen in the rounded edges of the illuminated area. Similar to the quadrant configuration, each circular segment produces an image that is concatenated with the consecutive image obtained for the same circular segment, to transfer a total of 8 concatenated images into one.

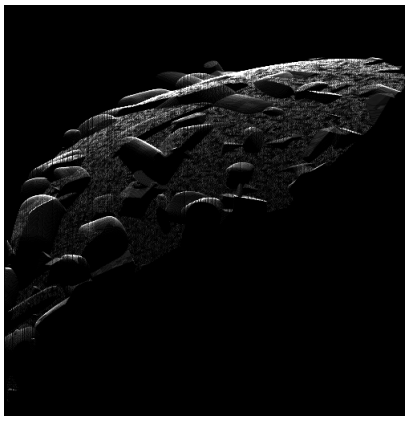


Fig. 4.11 Image generated using the information from a single circular segment.

4.4 Results and discussion

In this section, results of validating the use of the sonar FoV portions described above are presented.

4.4.1 Comparison of the FoV portions using simulated data

To evaluate the use of different FoV portions detailed in section 4.3, for each technique, a data set with 1000 pairs of images/vectors is generated with their motion labels in the three DoFs. The data set is split into 950 training samples and 50 validation samples. The following notation will be used:

- **Whole FoV:** Images and vectors are generated using all the rays in the FoV.
- **Single side:** Images and vectors are generated using the rays of one side of the vehicle. The selected side corresponds to the segment of the FoV from -50° to 50° in aperture and from -50° to 0° in elevation as shown in Fig. 4.6.
- **Single quadrant:** The rays from the Q1 quadrant are used to generate the images and vectors of the data set.
- **Four quadrants:** All four quadrants are used to generate the images and vectors.

- **Circular segments:** Circular segments from all four quadrants are used to generate the images and vectors.

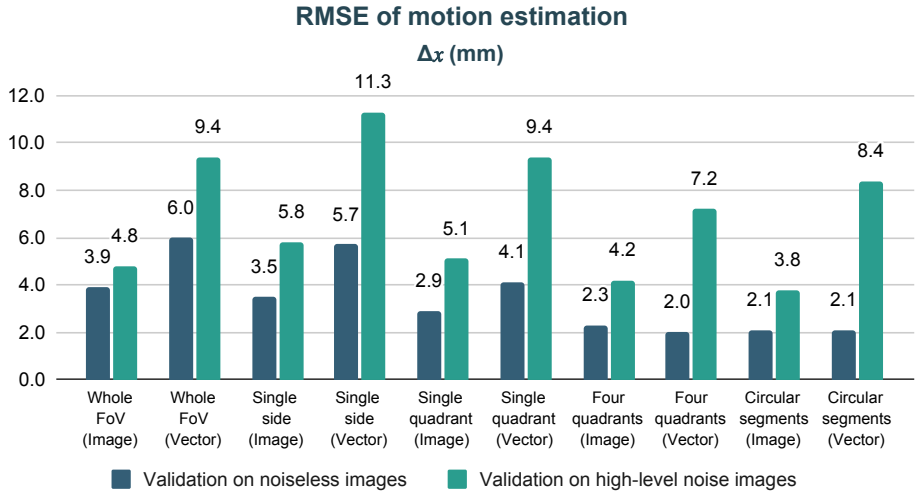
The RMSE for the motion estimation is obtained individually for each of the 3 DoF. The DL networks are trained using noiseless images. However, the validation is based either on noiseless images or on images with a high-level noise added to the pixels. Following the considerations specified in subsection 2.2.3, the noise is added with parameters: (i) the Gaussian distribution has a mean and standard deviation of 13.7% and 3.1% of the maximum pixel value, respectively; (ii) the Rayleigh distribution has a scale parameter of 13.7% of the maximum pixel value.

Every time a PoseNet and a PoseNetVec are trained for all the configurations, the following setup is considered. The learning rate at the start of the training is set to 10^{-4} and it is reduced to 0.5×10^{-4} at epoch 12. At epoch 16, it is reduced to 10^{-5}. The training is stopped at epoch 24 or when the validation loss converges. The Adam optimization algorithm is used. The MSE loss function is given by equation 3.1. For the experiments in this chapter, S and W are set to 4 and 3, respectively.

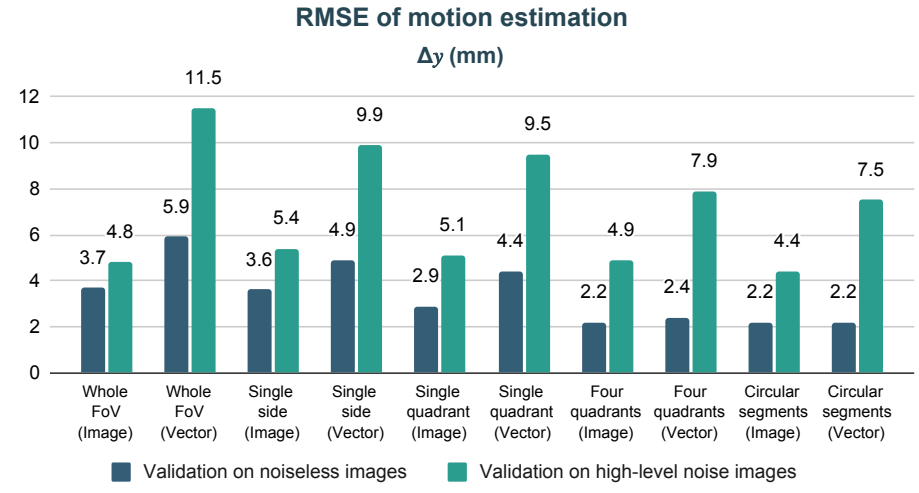
Figs. 4.12a, 4.12b, and 4.12c show the RMSE of the estimates for the displacement in x and y directions and rotation around z-axis, respectively. The RMSE are shown in pairs, the blue bars correspond to validation with noiseless images and the green bars correspond to validation with the high-level noise images. Also, Table 4.3 presents the computation time required to obtain one motion estimate. The time values in the table are averaged over 50 measurements for each of the DL networks. The comparison is done for a standard PC with i5-6500 CPU @3.0GHz processor without a GPU, and 8.0 GB of RAM.

From Fig. 4.12 and the computation times in Table 4.3 and focusing on validation with noiseless images only, the following observations are made:

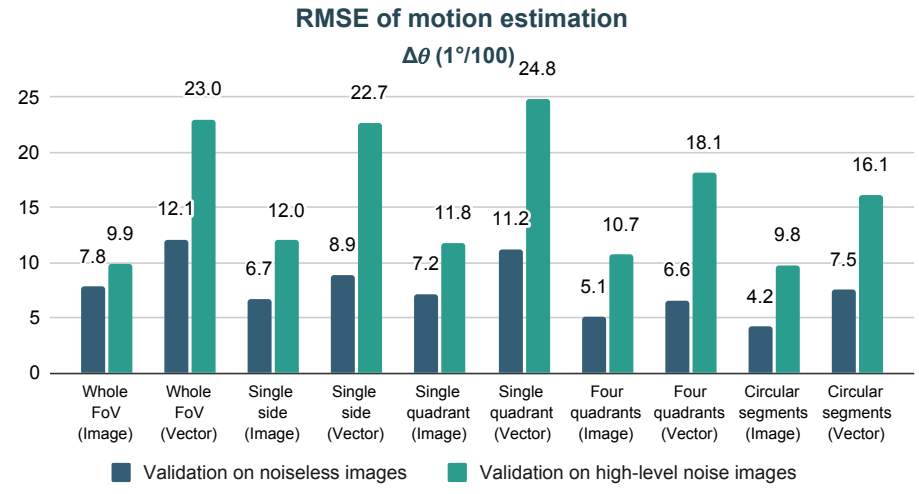
- For any FoV configuration, the image technique has a better estimation accuracy than the corresponding vector technique. However, the difference is not high, and, for the best estimation techniques using data from the four quadrants, the difference is negligible.



(a)



(b)



(c)

Fig. 4.12 Validation RMSE for the motion parameters Δ_x, Δ_y and Δ_θ, shown in (a), (b) and (c), respectively.

Configuration	DL network time (ms)
Whole FoV (Image)	160
Whole FoV (Vector)	16
Single side (Image)	139
Single side (Vector)	17
Single quadrant (Image)	65
Single quadrant (Vector)	13
Four quadrants (Image)	309
Four quadrants (Vector)	27
Circular segments (Image)	307
Circular segments (Vector)	29

Table 4.3 Average computation time required by the DL networks for one measurement.

- In general, the motion estimation in x and y directions show a similar accuracy. This can be due to the same size of the FoV in the aperture and elevation dimensions.
- Splitting the whole FoV into smaller portions results in improvement of the estimation accuracy.
- The vector techniques require an order of magnitude smaller computation time to produce the measurement than the corresponding image technique.
- The lowest estimation error is obtained with the configurations of 4 quadrants (for both the rectangular or circular segmentation). The image and vector techniques show an RMSE for translation in either x or y directions slightly above 2 mm and for rotation between 0.042° and 0.051°.
- Using the FoV for one side of the vehicle is preferable over both sides. It helps to avoid the ambiguity mentioned in subsection 4.3.3.
- The single quadrant configuration further improves the accuracy in estimation of the translations, but not the rotation.
- The four quadrant configuration shows better estimation accuracy compared to the single quadrant and single side configurations, at the cost of a higher computation time.

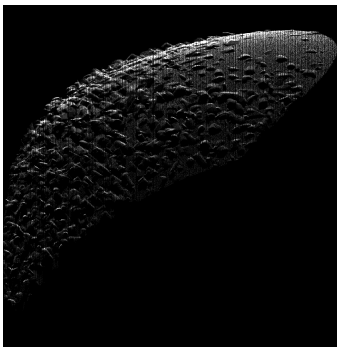


Fig. 4.13 Image generated using the information from only one circular segment at a distance of 10 m from the seabed.

- The circular segmentation provides results similar to the rectangular segmentation, despite some loss of information due to removing data from the corners of the quadrants.

When validating with noisy images, the motion estimation accuracy presents higher error than the validation with noiseless images. This is expected, given that the networks are trained using noiseless data for both types of validation. However, the obtained RMSE that corresponds to each FoV configuration shows the same trend as in the case of validation with noiseless data described in the points above.

4.4.2 Results for an increased distance of the sonar vehicle from the seafloor

To investigate how distance can affect the accuracy of the motion estimation, a new data set of 1000 samples is generated using the circular segments in the four quadrants. The difference is that the distance of the sonar to the seafloor is now 10 m rather than 2.5 m. An example of an image obtained with this distance from the seafloor is shown in Fig. 4.13. The pixels in the first row of the image correspond to a range of 9 m from the sonar and the pixels in the last row corresponds to the range of 24.5 m.

Additionally, to reduce the estimation error for validation with noisy images, the image and vector techniques are trained with a data set containing noisy images. For comparison, the training with noisy images was done using the data sets for the cases of 2.5 m and 10 m

heights. The noisy images have a high-level noise with the same parameters as described in subsection 4.4.1.

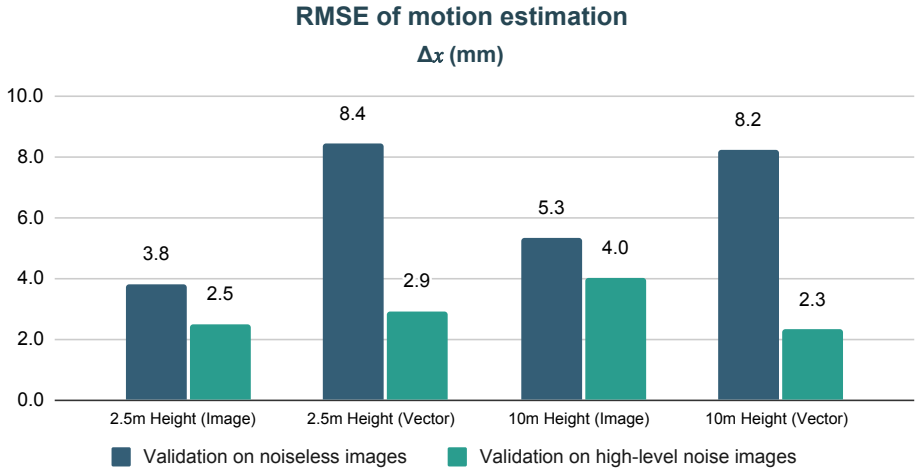
Figs. 4.14a, 4.14b, and 4.14c show validation results for training with noiseless and noisy data at two different heights from the seafloor. It can be seen that the RMSE is significantly reduced when training with high-level noise images compared to training with noiseless data and validating with noisy data. The obtained RMSE is comparable with the case of training and validating with noiseless data for the image and vector techniques. The results give a better idea of what is the approximate error when using real data, given that real images are noisy. Regarding the results of having the sonar at 2.5 m and 10 m from the seafloor, the distance does not affect significantly the estimation accuracy.

4.4.3 Dependence of the RMSE on the range quantization

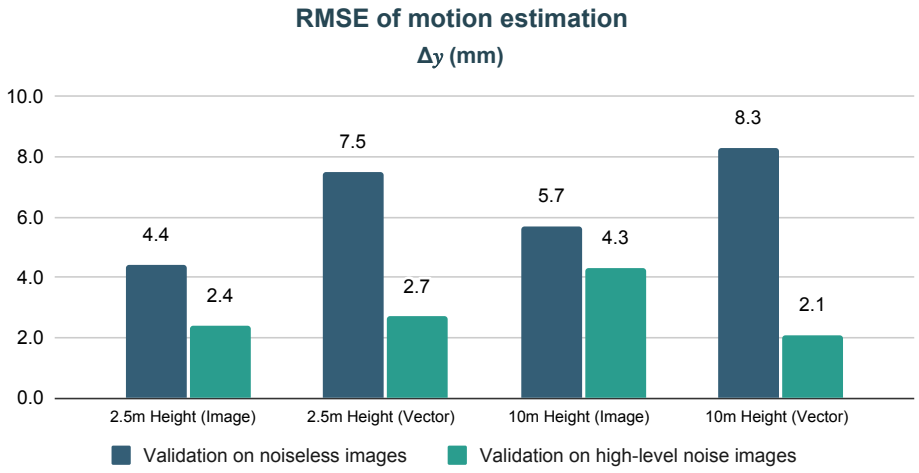
The number of range levels (quantization levels) in the vectors can be tuned to improve the estimation accuracy. For this, data sets with different numbers of range levels are generated. When generating the images, the intensities collected by the sonar are spread along range dimension of the image. The chosen number of range levels are a power of 2 (16, 32, ..., 512, and 1024). For validating each number of range levels, a data set of 1000 samples is generated and used to train a PoseNetVec. The data sets are generated for the sonar at a height of 10 m and the circular segment configuration.

Fig. 4.15 presents the RMSE obtained using different number of range levels to generate the images. The curves show that for the vector technique, the best cases are with 128 and 256 range levels, where the error is minimized. The increase of the RMSE while decreasing the number of levels can be due to the poor range resolution with a small number of levels. This is equivalent to adding some noise to the data. The RMSE increase with an increased number of levels can be caused by the large number of internal parameters to be trained by the DL network.

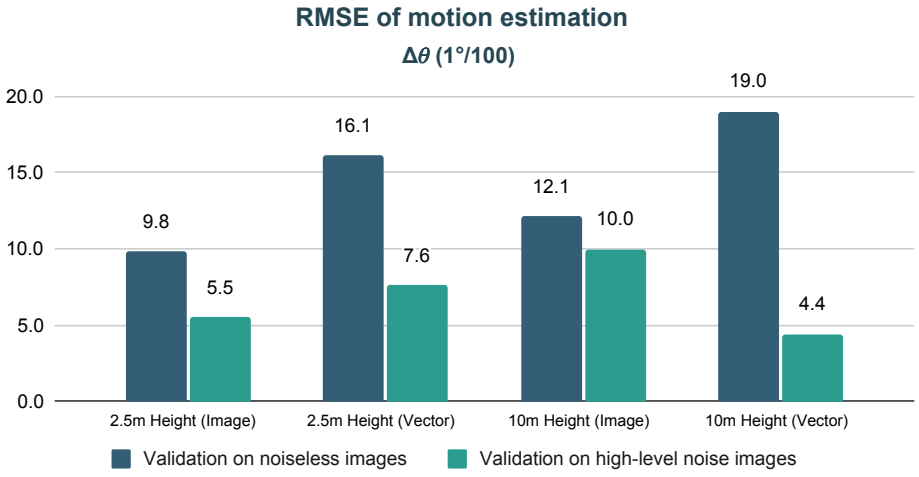
In Chapter 3, for the case of noiseless images, an RMSE of around 1 mm is achieved in the direction the FLS is looking, but in the sideways direction the RMSE is always higher than 2.3 mm. The vector technique proposed here with 128 and 256 range levels provides



(a)



(b)



(c)

Fig. 4.14 Validation RMSE when training with noiseless images and high-level noise images and validating with noisy images for the motion parameters Δ_x, Δ_y and Δ_θ, shown in (a), (b) and (c), respectively.

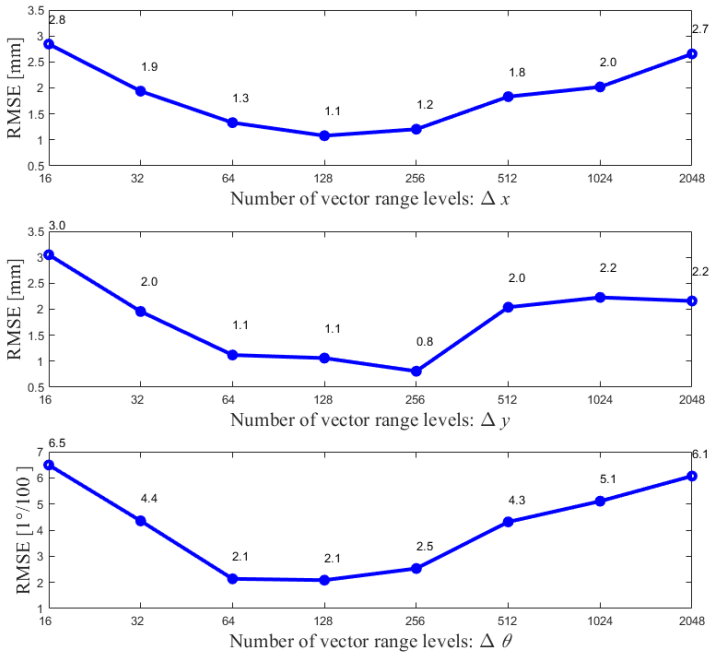


Fig. 4.15 RMSE obtained for each motion estimation parameter against the number of range levels for the vector technique with the circular segmentation. The vehicle's distance from the seabed is 10 m. The training and validation are using noiseless images.

the RMSE for the translation motion (x and y) between 0.8 mm and 1.2 mm, which is an improvement in the estimation accuracy.

For the rotation, the RMSE in Chapter 3 in the best case is 0.047°. This is higher than the RMSE obtained by the vector technique, which is around 0.025°, which is again an improvement in the accuracy. These results show that the vector technique can provide a high estimation accuracy. The great advantage of the vector technique is reduction in the complexity of the DL network. This allows a high processing speed and makes the technique suitable for real-time applications.

4.4.4 Trajectory reconstruction using simulated data

As described in Chapter 3, a trained DL network can be used to estimate the displacements between each pair of consecutive images. With these estimates, the vehicle trajectory can be recovered. The points of the trajectory are represented as x_i and y_i and the orientation of the

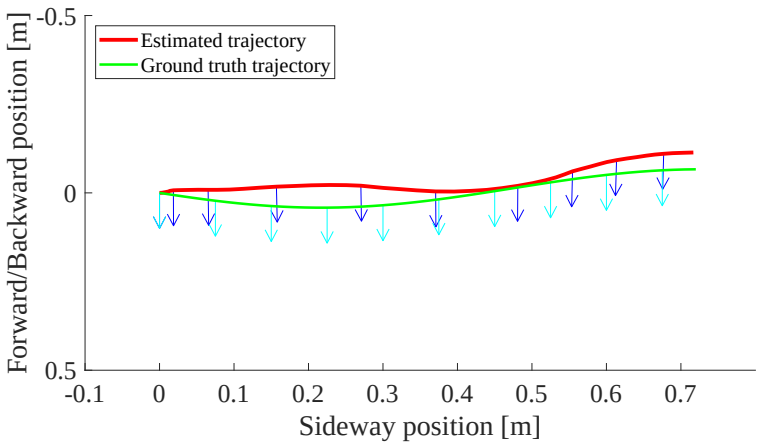


Fig. 4.16 First simulated trajectory (256 range levels): Ground truth trajectory (green line), ground truth orientation (cyan arrows), estimated trajectory (red line), estimated vehicle orientation (blue arrows) using 50 pairs of vectors of simulated data.

vehicle is represented as θ_i, where i is an index that refers to the positions where the images are obtained. The initial position and orientation x_1, y_1 and θ_1 are set to zero and the other points are calculated using the equation 3.4.

Two examples of trajectories generated with simulated data are shown in Figures 4.16 and 4.17. Both trajectories have most of the motion in sideways direction with no rotation. The motion estimates are done using the vector technique. The vehicle height from the seafloor is 10 m as in the cases described in the previous subsection with two numbers of range levels. The first trajectory is made of 51 vehicle positions (50 pairs of vectors for estimation) and the number of range levels is 256. The second trajectory is made of 55 positions using vectors of 512 range levels.

It can be seen that both estimated trajectories follow the ground truth trajectory with most of the motion in sideways direction. The recovered trajectory for the case of 256 range levels seems smoother when the vehicle moves backwards almost at the end of the trajectory, whilst the recovered trajectory presents abrupt changes in the same part of the trajectory. The better performance using 256 levels was expected, given that the motion estimation with 256 levels is slightly more accurate than the case of 512 as shown in Fig. 4.15.

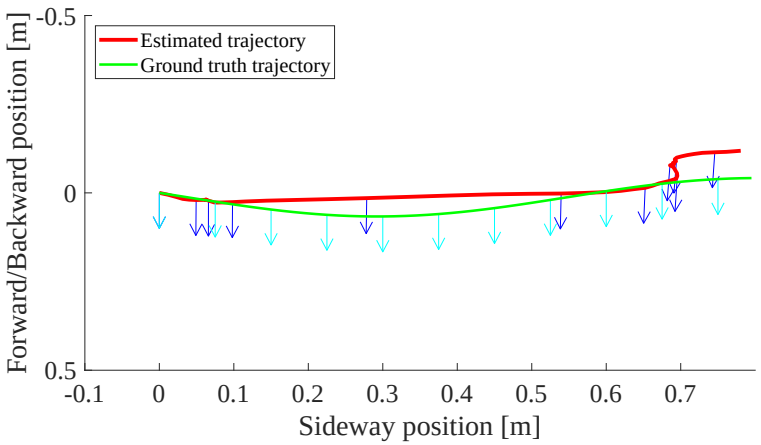


Fig. 4.17 Second simulated trajectory (512 range levels): Ground truth trajectory (green line), ground truth orientation (cyan arrows), estimated trajectory (red line), estimated vehicle orientation (blue arrows) using 54 pairs of vectors of simulated data.

4.4.5 Trajectory reconstruction and mosaic building using real data

The accuracy of the vector technique is evaluated using real data. Therefore the data set obtained by a real sonar sensor as described in subsection 3.2.3 is used. Furthermore, the data set generated from a simulated environment that resembles a ship's hull described in subsection 3.2.2 is used for training. The images have the same aperture and elevation angles as images obtained by the DIDSON sonar.

The DIDSON sonar is an FLS, therefore the vertical axis on the image corresponds to the forward motion of the sonar and the horizontal axis corresponds to the sideways motion. To train the network, noise is added to the images given that the real images are noisy. The noise parameters have been adjusted to give the best estimates with the real data set and they are as follows: For the Gaussian distribution the mean and standard deviation are set to 4% and 2% of the maximum pixel value, respectively. And for the Rayleigh distribution, the scale parameter is set to 4% of the maximum pixel value.

The images obtained from the DIDSON sonar cannot be taken from 4 quadrants and the aperture and elevation angles are different from the data sets used in previous subsections. Therefore, the images are modified by separating them in azimuth in two halves, where each

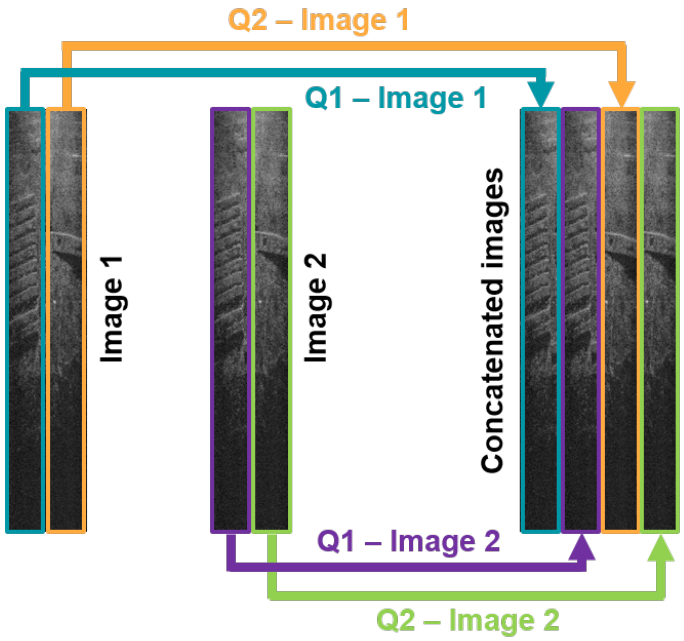


Fig. 4.18 Each of two sonar images is split into two parts, representing two quadrants. The quadrants are re-ordered as shown and are converted into vectors to be used as the input of the DL network.

half represents a quadrant. Using pairs of consecutive images, the input of the PoseNetVec is the concatenation of 4 vectors of size 512. The first 2 vectors correspond to the left columns of the first and second images and the other 2 vectors correspond to the right columns of the images as shown in Fig. 4.18.

In Fig. 4.19, the full estimated trajectory and orientation of the vehicle are displayed with red line and blue arrows, respectively. According to the reconstructed trajectory, a mosaic is built by merging the 520 images in the data set. When the images are combined, the pixel intensities are averaged in the overlapping images. It can be seen that during the ship's hull inspection, the sonar mostly moves in the sideways direction with small motions in the forward/backward direction and rotation. This can be seen in the estimated trajectory, while the small rotation can be seen pointing forward most of the time.

The estimated trajectory and mosaic look accurate compared to similar pictures in other works [29, 79]. Furthermore, the time to obtain the motion estimates using the computer described in subsection 4.4.1 was about 12 ms per estimate. This computation time is

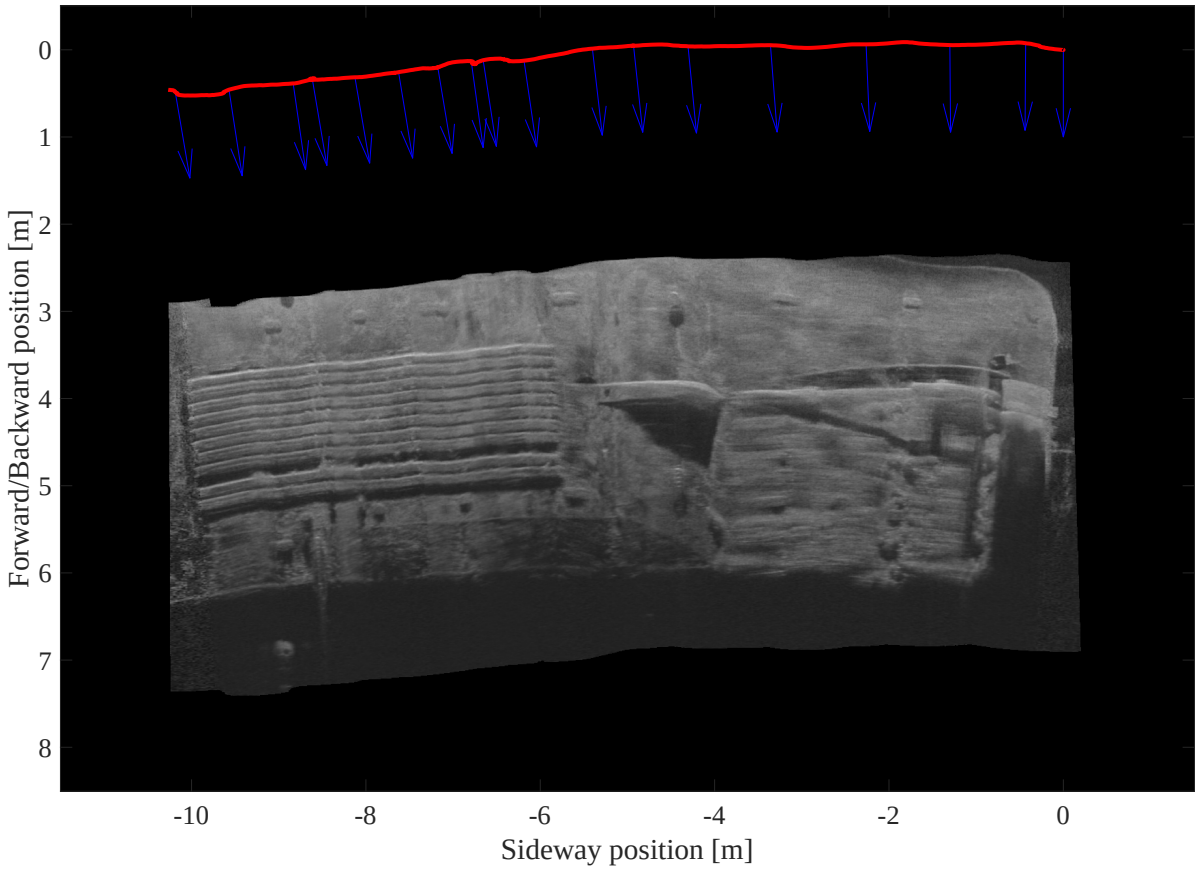


Fig. 4.19 Reconstructed vehicle trajectory (red line), vehicle orientation (blue arrows) and mosaic built using 520 sonar images of the ship's hull real data. From the mosaic, multiple parts of the ship's hull are clearly recognizable, like the sacrificial anodes, the propeller and the keel.

considerably smaller than for the method in [79] and the DL method from Chapter 3 (sonar images without preprocessing before the DL network). The computation times for each pair of images are 25.69 s and 56.9 ms, for the method in [79] and the method from Chapter 3, respectively. Moreover, the computation time for the vector technique is much smaller than the time that it takes for the DIDSON sonar to generate an image with a frame rate of 21 fps (an image every 48.62 ms). This shows that the vector technique is suitable for accurate motion estimation in real-time.

4.5 DL network to estimate height from the seafloor

A method to use a DL network to estimate the height of an underwater vehicle from the seafloor is proposed. For this, a minimum and maximum height are considered. The training data set is generated by the simulator by placing the sonar at a random value for height value with a uniform distribution between the minimum and maximum heights. Then an image is generated by the sonar and the corresponding label (height in the moment of generating the image) is attached to the image to produce a training sample. Two different simulated underwater types of scenario are used to generate data sets. The data sets are described as follows:

- **Rocky field:** This data set is generated from the simulated rocky field underwater scenarios described in subsection 3.2.1. The minimum and maximum heights are 1 m and 10 m, respectively. The looking down sonar configuration (whole FoV) from Fig. 4.3 is used to generate the images. The image size is 512×2000 pixels. The data set consists of 1250 images without noise and their corresponding label.

- **Ship's hull (Noiseless):** This data set is generated from the simulated ship's hull underwater scenarios described in subsection 3.2.2. In this case the distance to from the sonar to the ship's hull is measured instead of the height from the seafloor. The minimum height is 20 cm and the maximum height is 55 cm. The sonar configuration shown in Fig. 3.3b is used to generate the images. For this configuration the sonar is looking to a side of the vehicle with pitch angle (α) of 4°. The FoV is the same for

DIDSON 300 sonar sensor. The image size is 512×96 pixels. The data set is made of 3000 images without noise.

- **Ship's hull (Noisy):** This data is the same as the previous one with the difference that the images are noisy. The noise model uses the Gaussian distribution with a mean and standard deviation of 13.7% and 3.1% of the maximum pixel value, respectively; and the Rayleigh distribution with a scale parameter of 13.7% of the maximum pixel value.

Each data set is divided into 95% and 5% for the training set and validation set, respectively. The PoseNet is adapted to perform the height estimation. The parameters of the network are shown in Table 4.4, where values in the column "Output size per channel" depend on the input image size specified above. For each data set, the image and vector techniques are used and compared. The table only shows the output size for the image technique.

Layer	**Kernel**	**Channels**	**Stride**	**Output size per channel**		**ReLU & Batch norm**
				Rocky field data set	**Ship's hull data sets**	
Conv1	7×7	32	2×2	256×1000	256×48	Yes
Conv2	5×5	64	2×2	128×500	128×24	Yes
Conv3	3×3	128	2×2	64×250	64×12	Yes
Conv4	3×3	256	2×2	32×125	32×6	Yes
Conv5	3×3	512	2×2	16×63	16×3	Yes
Conv6	3×3	512	2×2	8×32	8×2	Yes
Conv7	3×3	512	2×2	4×16	4×1	Yes
Conv8	3×3	1024	2×2	2×8	2×1	Yes
Conv9	3×3	1024	2×2	1×4	1×1	No
Average	4×4	1024	1×1	1×4	1×1	No

Table 4.4 Parameters of PoseNet and PoseNetVec layers used for height estimation.

Table 4.5 shows the RMSE and the ratio RMSE / training interval for height estimation for the three types of data set for the image and vector techniques. It can be seen that the estimation error is very similar regardless the image or vector technique is used. For the rocky field data set, which has a large height interval of 9 m, the ratio is smaller than the others two data sets. This can be due to the looking-down orientation of the sonar that seems

to favor the height estimation. This orientation can give direct information about the distance to the seafloor. For the ship's hull data sets, the noisy images data set shows better estimates than noiseless images data set. This can be because the network that is trained with noisy images tends to ignore small groups of pixels that are not representative of the real distance to the ship's hull, and they are just part of an object that slightly appears in the image and seems to be isolated from the rest of objects. For example the bright pixels in the centre of the images in Fig. 4.20.

Data set	**RMSE** ***(cm)***	**RMSE / height interval** ***(%)***	**RMSE** ***(cm)***	**RMSE / height interval** ***(%)***
	Image technique		**Vector technique**	
Rocky field	11.91	1.32	12.17	1.35
Ship's hull (Noiseless)	1.20	3.43	1.20	3.43
Ship's hull (Noisy)	0.78	2.23	1.01	2.89

Table 4.5 Validation RMSE for height estimation.

4.6 Conclusions

Two main contributions for underwater motion estimation are presented in this chapter. The first contribution is a technique that consists in converting sonar images into vectors by adding up the pixels in each row. The use of vectors reduces the size of the DL networks that are used to estimate the motion. The second contribution is the division of the beams in the FoV of a simulated sonar into four groups (quadrants). An image is generated from each group. This is combined with the vector technique by converting the images into vectors and grouped together as the input of the DL network. The network is trained using simulated data. The proposed vector technique using the four images shows better accuracy and faster computation times compared to a similar method from the literature and the DL method from Chapter 3 which uses the sonar images without any kind of preprocessing before applying the DL network.

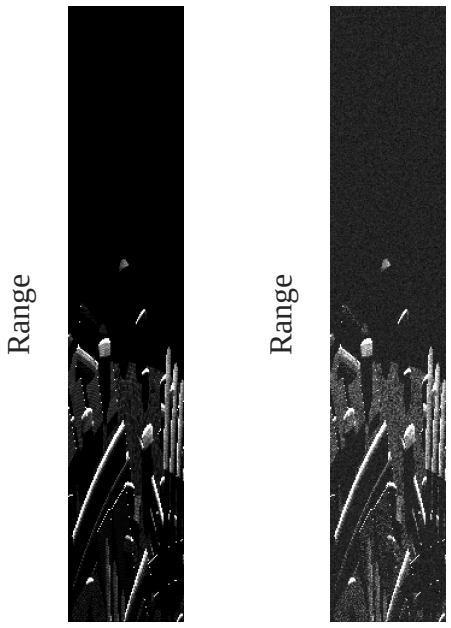


Fig. 4.20 Examples of sonar images generated from the ship's hull data set with groups of pixels from a portion of an object isolated from the rest of the objects. (a) Image without noise. (b) Image with noise.

Validation of the vector technique with simulated and real data is presented. For the case of real data, sonar images from an FLS are used by adapting the quadrants with the vector technique to the features of the images. The application on real data shows good results and the potential to perform low-cost and fast motion estimation in real time.

These results suggest that it might be possible to design a motion estimation system with a few hydrophones possessing a low space directivity. In Chapter 5 such designs are investigated and compared with techniques presented so far.

The height estimator presented in this chapter shows satisfactory results with estimation errors below 3.5%. The proposed technique uses the same information that the motion estimation techniques. Therefore, reusing data and a reduction in the equipment costs is possible when both systems are implemented. Furthermore, a height estimator can be used

to optimize the DL network and to quickly switch to different DL networks trained with different sonar height configurations.

Chapter 5

Motion estimation using impulse responses from the seafloor

In previous chapters, high navigation accuracy in motion estimation has been achieved using sonar images (or compressed images) augmented with DL networks. The sonar images are estimates of the magnitudes of channel impulse responses sampled in time (with a ping period) representing the underwater acoustic environment. More information about the environment is contained in (almost) continuous-time estimates of the channel impulse responses. Such estimates can be obtained using full-duplex technology [149, 150]. Rather than using sonar images, the use of channel impulse response estimates for underwater vehicle motion estimation is investigated in this chapter. In Section 5.1, an introduction to the impulse responses technique and the difference between the method and a CVL are presented. In Section 5.2, modifications of the simulator from Chapter 2 to generate channel impulse responses and corresponding data sets are described. In Section 5.3, the DL network for motion estimation using channel impulse responses is described. Section 5.4 presents results of motion estimation experiments for numerous sets of system parameters. A comparison with methods that use sonar images is also presented. Finally, conclusions are given in Section 5.5.

The work in this chapter is presented in the paper: J. E. Almanza-Medina, B. Henson, L. Shen, and Y. V. Zakharov, "Motion estimation of underwater platforms using impulse responses from the seafloor," *IEEE Access*, vol. 10, pp. 127 047–127 060, 2022.

5.1 Introduction

In Chapter 3, a DL network was applied to sonar images to estimate the motion of an underwater vehicle, then in Chapter 4 a complexity reduction was achieved by compressing the sonar images. In this chapter, instead of using sonar images for motion estimation, a novel technique that uses channel impulse responses is proposed. Similar to a CVL, and considering the motion on a plane parallel to the seafloor, a single transmitter is used to send acoustic signals to the seafloor and several (up to four in the simulated scenarios) receivers (hydrophones) placed around the transmitter to listen to the reflected signals.

One of the main differences between the proposed technique and a CVL is that the CVL works by transmitting two short ping pulses that are reflected from the seafloor and the echoes are measured at the receivers [65, 151], while the proposed technique uses a continuous signal transmission and the receivers are continuously listening at the same time. This allows continuous measurement of the channel impulse response while systems like the CVL have a ping period for each measurement, which limits amount of information used for the motion estimation. The simultaneous transmission/listening operation is possible to implement with full-duplex techniques such as those recently proposed in [149, 150]. Another difference is that a CVL measures the correlation between responses of the two pings [151], whereas the proposed technique exploits the complex-valued channel impulse response variation in time and uses a DL network to estimate the motion. Finally, a CVL requires an array with a large number of receivers [65, 152] while the proposed technique can achieve fairly accurate motion estimation with only two receivers and high accuracy with four receivers. Compared to the techniques based on environment referencing, such as in [29, 79] and techniques from Chapters 3 and 4, where sonar images represent estimates of the channel impulse response magnitudes varying in time, and thus ignoring the phase information, the proposed method

is based on the whole information of the channel impulse response and thus is capable of providing a better motion estimation performance. The proposed technique is based on acoustic impulse response estimates with a much higher time resolution and carrying more information than the classical sonar images.

The motion is estimated by a DL network, which has as its input, two or more consecutive in time complex-valued baseband estimates of the impulse responses between the transmitter and hydrophones, obtained at consecutive time instances. However, a large amount of labeled data is required when training a DL network for regression tasks [127]. To solve this problem, synthetic data generated by a computer simulator is used, similar to the approach used in Chapters 3 and 4. The simulator can provide large data sets with the ground truth required for supervised learning.

In summary, a motion estimation method is proposed, based on DL analysis of the time variation of complex-valued channel impulse responses and enabled by full-duplex operation.

5.2 Simulator to generate impulse responses of the acoustic environment

This section describes how the simulator from Chapter 2 is adapted to generate channel impulse responses for complex underwater acoustic environments. These channel impulse responses are used for training the DL network for motion estimation and for evaluation of the proposed method.

As described in Chapter 2, simulators presented in the literature are capable of modeling underwater environments, vehicles and sensors. The sonar simulator presented in Chapter 2 is capable of simulating sonars that can be used for validating image registration techniques. The simulated sonar sensors and the underwater environment are implemented in Unity.

The aim of method presented in this chapter is to estimate the motion of a vehicle using relatively simple acoustic sensors (one projector and two - four hydrophones) and at the same time reduce the estimation complexity. The use of impulse responses allows us to achieve

both these aims. However, a simulator that can produce the channel impulse responses with a high precision is required.

The work [153] presents an underwater channel simulator that uses the BELLHOP ray-tracing model [154] to simulate an acoustic underwater channel with multipath propagation. A method to model signal transmissions in underwater acoustic communications is presented in [155], where the motion of the transmitter and the receiver is taken into account. This method reduces the model complexity with an approximation of the time-varying channel impulse response by sampling the trajectory and using local splines for the approximation. In [156] a communication channel simulator is presented. The simulator uses the Monterey-Miami Parabolic Equation model complemented with a linear surface model that simulates the sea surface motion. The work [157] complements the work [155] by reducing the simulation time even further. This is obtained with the use of baseband processing for modelling the signal transmission which allows a lower sampling rate compared to the passband processing.

For the proposed method, it is necessary to model the channel with a precision sufficient to recover the phases of the channel impulse response as opposed to sonar images, where only magnitudes are required. Another requirement is that the simulator must be capable of simulating multiple underwater objects to have a high variety of data for training and validation of DL networks. Also, the simulator must generate simulated scenarios in a 3D space. Therefore, the simulator from Chapter 2 is modified to generate impulse responses. The ray-tracing in this simulator provides a precision high enough for the signal phase to be taken into account and the Unity platform allows complicated 3D underwater environments to be generated.

The steps to generate the impulse responses are summarized in Fig. 5.1 and they are described in the next subsections.

5.2.1 Collecting data in a single position in the simulated environment

It is assumed that the transmitter and receiver are at different positions on the vehicle and they have (maybe different) predefined FoVs. A set of rays uniformly sampling the transmitter

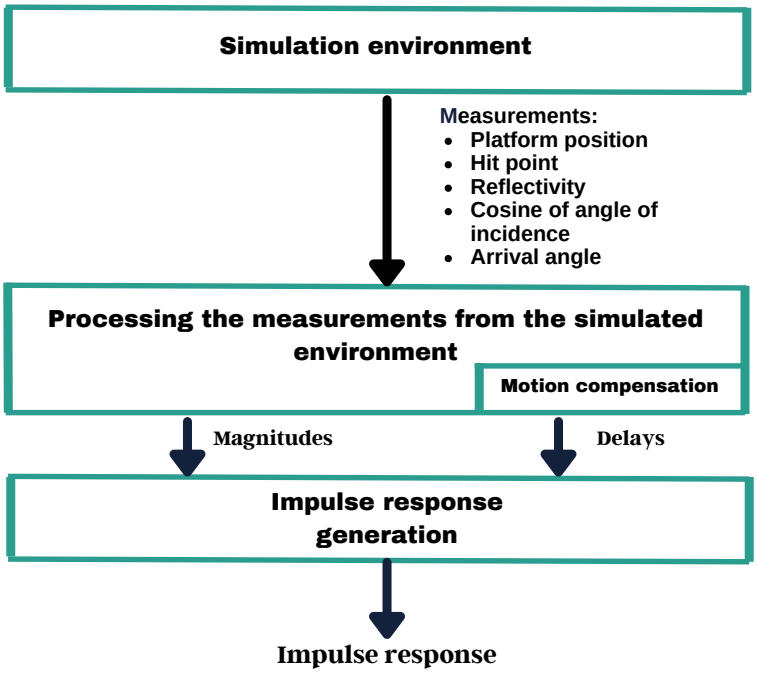


Fig. 5.1 Impulse response generation steps.

FoV are emitted from the transmitter position. When a ray hits an object, it bounces towards the receiver and the following measurements are recorded:

- **Vehicle position:** The position and orientation of the vehicle in the simulated scenario (global coordinates) when sending the rays. The position is represented by three coordinates, $A = [a_x, a_y, a_z]$.

- **Hit point:** For every ray, global coordinates of the point being hit by the ray from the transmitter. Every hit point is represented by three coordinates $P = [p_x, p_y, p_z]$.

- **Reflectivity (G_s):** A value between 0 and 1, where 0 corresponds to the case when the object totally absorbs the signal and it is not reflected and 1 corresponds to the case when the object reflects the signal without absorbtion.

- **Cosine of angle of incidence (J_s):** Cosine of the angle between a ray incident on a surface and the normal to the surface at the point of incidence.

- **Arrival angle:** Angle of the reflected ray arriving at the receiver. The angle is measured relative to the vehicle orientation.

5.2.2 Computation of the propagation delays for a static vehicle

The measurements obtained for every ray in a position of the vehicle are used to generate the corresponding impulse response. The rays outside the FoV of the receiver are discarded by analyzing the arrival angles. For each of the remaining rays, the distance from the transmitter to the hit point is calculated as

$$\delta_0 = ||P - A||, \tag{5.1}$$

and the distance from the hit point to the receiver is calculated as

$$\delta_1 = ||B - P||, \tag{5.2}$$

where $B = [b_x, b_y, b_z]$ is the position of the receiver on the vehicle. Considering that a receiver is placed at point B and it has an offset $E = [e_x, e_y, 0]$ from the transmitter position, then

$$B = [a_x + e_x, a_y + e_y, a_z]. \tag{5.3}$$

The propagation time that a ray takes to travel over the distance δ_0 + δ_1, from the transmitter to the receiver, assuming that the vehicle is static is given by

$$\tau_P = \tau_0 + \tau_1 = \frac{\delta_0}{c_{\text{uw}}} + \frac{\delta_1}{c_{\text{uw}}}, \tag{5.4}$$

where c_{uw} is the speed of sound underwater, considered to be 1500 m/s. The calculated times for each ray are saved into a vector of propagation delays **D**.

In addition, a vector **I** is defined with ray magnitudes computed as the product $J_s G_s$.

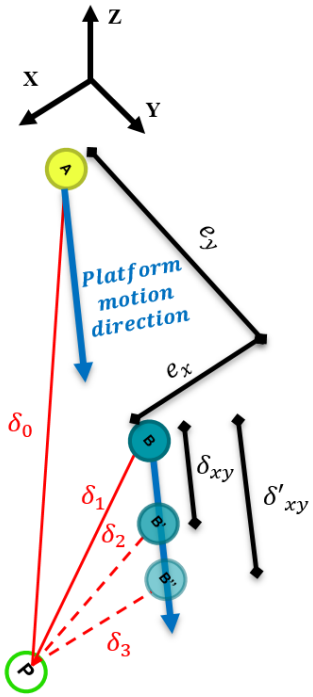


Fig. 5.2 Diagram that explains the delay compensation due to the vehicle motion. The point B moves with the vehicle while the ray travels from the hit point to the receiver.

5.2.3 Motion compensation

The propagation delay τ_P in (5.4) is calculated assuming that the vehicle is static during the travel of the rays. However, for a more accurate representation of the impulse response, the delay computation should take into account that the vehicle is moving at a certain velocity $V = [V_x, V_y]$. The diagram in Fig. 5.2 shows the motion of the vehicle and how it affects the propagation delay of a ray.

The displacement of the receiver ($\delta_{xy} = [\delta_x, \delta_y]$) at velocity V during the time that the ray travels can be approximately calculated as

$$\delta_{xy} = (\tau_0 + \tau_1)V. \tag{5.5}$$

The propagation delay should be calculated at the displaced position of the receiver, which is

$$B' = [a_x + e_x + \delta_x, a_y + e_y + \delta_y, a_z]. \tag{5.6}$$

The distance from the hit point to the displaced position of the receiver is given by

$$\delta_2 = ||B' - P||, \tag{5.7}$$

and the time to travel the distance δ_2 is computed as

$$\tau_2 = \frac{\delta_2}{c_{\text{uw}}}, \tag{5.8}$$

then the propagation time along the ray taking into account that the vehicle is moving is

$$\tau'_P = \tau_0 + \tau_2. \tag{5.9}$$

The delay τ'_P in (5.9) is a more accurate ray propagation delay compared to the delay τ_P computed in (5.4). However, this computation can be further refined by repeating the steps (5.5) to (5.9) as follows. The propagation delay of the ray is adjusted once again by taking into account the updated propagation time τ'_P. The receiver moves to a new position B'' during this time. The distance from the hit point to the receiver at B'' and the time that a ray travels this distance are

$$\delta_3 = ||B'' - P||, \tag{5.10}$$

$$\tau_3 = \frac{\delta_3}{c_{\text{uw}}}, \tag{5.11}$$

where

$$B'' = [a_x + e_x + \delta'_x, a_y + e_y + \delta'_y, a_z], \tag{5.12}$$

$$\delta'_{xy} = [\delta'_x, \delta'_y] = (\tau_0 + \tau_2)V. \tag{5.13}$$

The propagation time along the ray taking into account that the vehicle is moving is refined as

$$\tau_P'' = \tau_0 + \tau_3. \tag{5.14}$$

This process can be iterated further, but for this work, the delay is compensated only twice given that further adjustment of the delay is negligible for the delay accuracy required. Finally, the vector **D** is populated by the updated propagation delays τ_P'' and used together with the vector **I** to generate the impulse response.

5.2.4 Impulse response generation

The delay **D** and magnitude **I** of all rays (multipath components) are used for computing the channel impulse response. The maximum propagation delay is $\tau_{\max} = \max\{\mathbf{D}\}$. The carrier frequency is f_c, the frequency bandwidth of interest is F_d, thus the frequency interval of interest is $[f_c - F_d/2, f_c + F_d/2]$.

First, the channel frequency response is computed and then converted into the time domain impulse response ($\mathbf{h}$) using the inverse Fast Fourier Transform (IFFT). The frequency response $H_f(k)$ at a frequency f_k is computed as:

$$H_f(k) = \begin{cases} \sum_{n=0}^{N-1} I_n e^{-j2\pi D_n(f_c+f_k)}, & f_k \in [-a, a] \\ 0, & f_k \in [-b, -a) \cup (a, b] \end{cases} \tag{5.15}$$

where $f_k = k\Delta f$, $\Delta f < 1/(2\tau_{\max})$ is a frequency step, $a = (1+\alpha)F_d/2$, $b = f_d/2$, N is the number of multipath components (number of values in **D**), I_n and D_n are the real-valued magnitude and delay of the nth multipath component, respectively, the IFFT size is $f_d/\Delta f$, f_d is the sampling rate, and α is the roll-off factor of the raised cosine function $X_R(k)$ (see Fig. 5.3) defined as:

$$X_R(k) = \begin{cases} 1, & f_k \in [-c, c] \\ \frac{1}{2}\Big[1 + \cos\Big(\frac{2\pi|f_k|}{\alpha F_d} - \frac{(1-\alpha)F_d}{2}\Big)\Big], & f_k \in [-a, -c) \cup (a, c], \\ 0, & f_k \in [-b, -a) \cup (a, b] \end{cases} \tag{5.16}$$

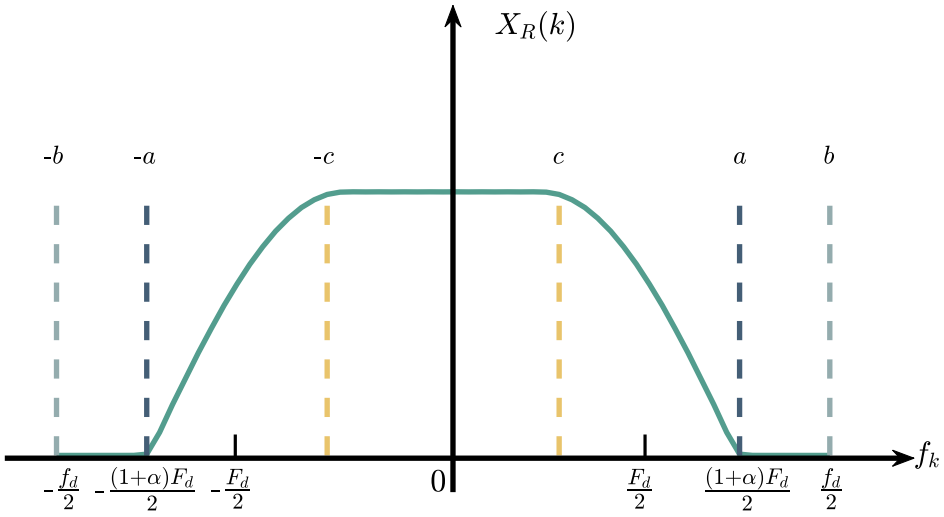


Fig. 5.3 The raised-cosine spectrum with roll-off factor α.

where $c = (1-\alpha)F_d/2$. The frequency response is multiplied by $X_R(k)$ to guarantee a smooth transition at the edges of the frequency bandwidth as:

$$H(k) = H_f(k)X_R(k), \quad f_k \in [-b, b]. \tag{5.17}$$

The magnitude, real part and imaginary part of a single impulse response ($|\mathbf{h}|$, $\mathrm{Re}(\mathbf{h})$ and $\mathrm{Im}(\mathbf{h})$, respectively) are displayed in Fig. 5.4. To generate the impulse response, F_d is set to 4.8 kHz, f_c = 80 kHz and f_d = 8 kHz. The scenario where the impulse response was acquired is populated by objects with a height of 1 mm placed on the seafloor and the vehicle is at a height of 10 m. The horizontal axes in the figure represent taps, where one tap corresponds to 0.125 ms. It can be seen that the reflected signals from the majority of the objects are received with a propagation delay between 100 and 170 taps. It means that the objects are detected at distances between approximately 10 m and 15 m. It is important to note that in all the simulations presented in this chapter, perfect knowledge of the channel impulse responses is assumed. The way channel estimation errors affect the motion estimation performance is not investigated here and will be a subject of another work.

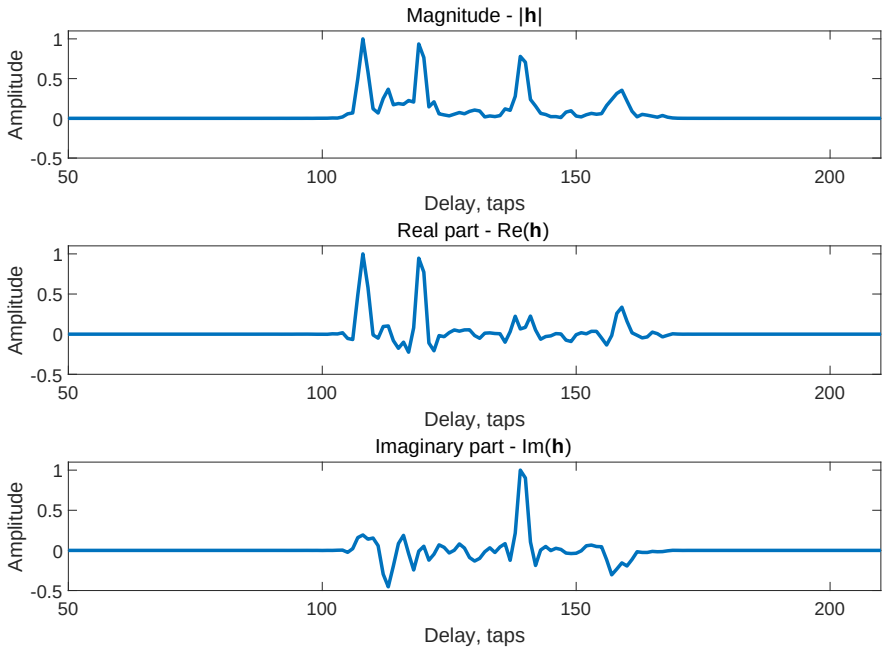


Fig. 5.4 Magnitude, real and imaginary parts of a single impulse response. The amplitudes are normalized.

5.2.5 Generation of data sets of impulse responses for training and validation of DL networks

Impulse responses for multiple receivers are generated and grouped together. It is assumed that a single transmitter is used to illuminate the underwater environment, then the difference in the impulse responses will depend on the positions of the receivers on the vehicle.

To generate a data set, groups of impulse responses from M receivers are packed together. The real and imaginary parts of the impulse responses ($\mathrm{Re}(\mathbf{h})$ and $\mathrm{Im}(\mathbf{h})$) from all the receivers are obtained. Compared to the techniques that use sonar images, where only magnitudes are obtained by the sonar, the advantage of using the complex values (real and imaginary parts) is that fast phase variations can be detected, which is an indication of motion.

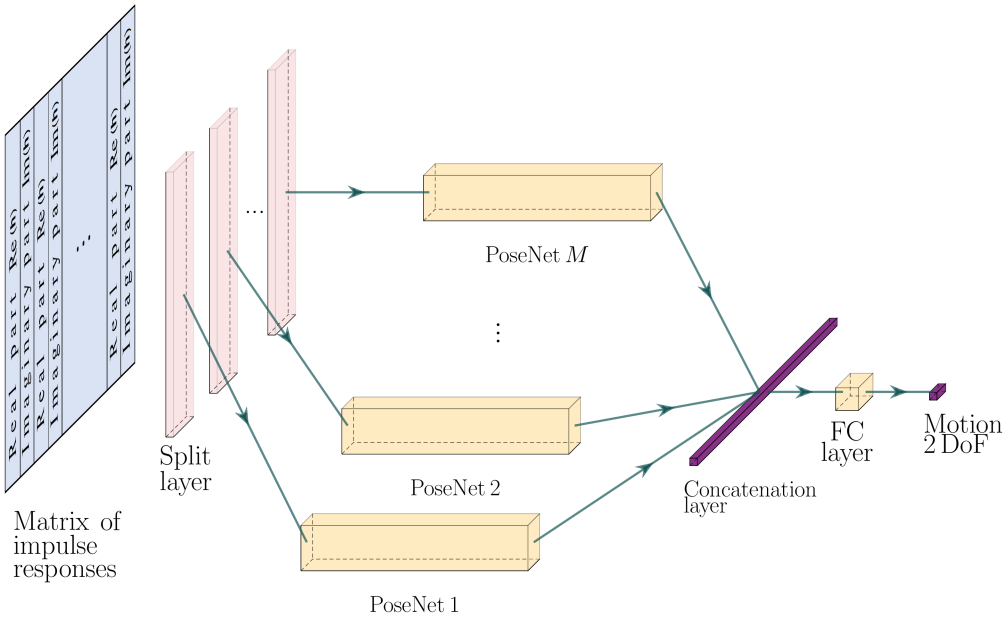
A matrix is made by putting together the real and imaginary parts of impulse responses generated in Q consecutive positions of the vehicle. Each column of a matrix is a real or

imaginary part of an impulse response. Therefore, each matrix has $2MQ$ columns. The order of the columns is defined as follows: (i) the real parts of the impulse responses from the first receiver at the Q vehicle positions; (ii) imaginary parts of the impulse responses from the first receiver at the Q vehicle positions; (iii) the same order (real parts and then imaginary parts) for the second receiver; (iv) and so on for the remainder of the receivers.

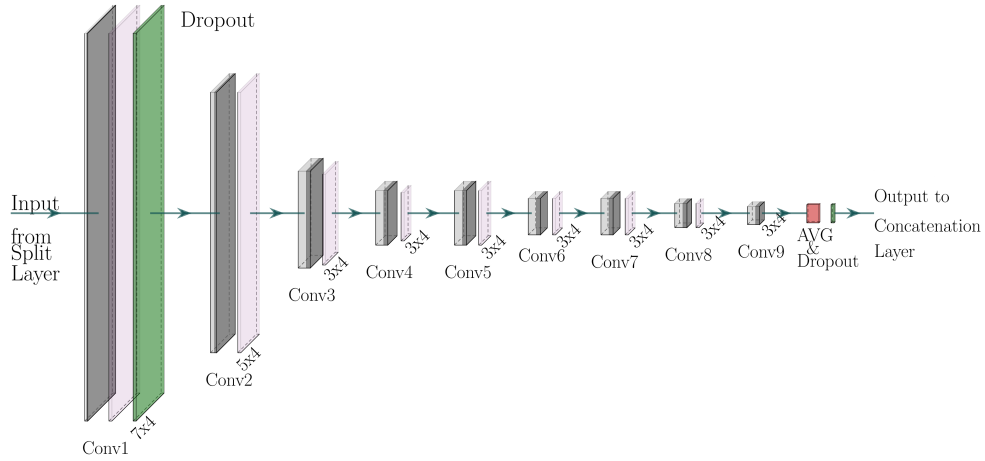
The maximum magnitude value is computed from the impulse responses of the M receivers in the Q consecutive vehicle positions whose real and imaginary parts are used to make the matrix. Then the matrix is normalized by dividing its elements by the maximum magnitude value. Every normalized matrix is associated with a label corresponding to the displacement between the first and the last positions. The displacement is denoted by $\Delta = [\Delta_x, \Delta_y]$ (motion in forward/backwards and sideways direction, respectively). Finally, a data set is generated with multiple matrices and their associated labels.

5.3 DL network

Numerous DL networks for motion estimation using sonar images are evaluated in Chapter 3 where the PoseNet showed the best performance. In this chapter, the PoseNet is adapted for motion estimation using the impulse responses. The whole network is shown in Fig. 5.5. The input of the network is a matrix with the real and imaginary parts of impulse responses obtained by M receivers in Q consecutive vehicle positions. The matrix is split into matrix blocks. Each matrix block corresponds to the data from one receiver. For every matrix block, the DL network contains a PoseNet, which is a series of convolutional layers as shown in Fig. 5.5b. The number of convolutional layers varies according to the need to optimize the network for each parameter configuration used to generate the impulse responses. It was observed that 9 convolutional layers was the maximum needed for the parameter configurations investigated in this work. All the convolutional layers with the exception of the last one use a ReLU activation function and batch normalization. An average pooling layer with an averaging window of size 4 is connected to the last convolutional layer. The



(a)



(b)

Fig. 5.5 (a) DL network with M PoseNets. (b) Architecture of PoseNet.

dropout regularization method is applied at the output of the first ReLU and to the average layer.

Parameters of the convolutional layers are presented in Table 5.1. Experiments showed better results in reducing overfitting when using dropout in the first and last layers only. The last layers of each PoseNet are concatenated together and connected to a fully connected layer and finally to an output regression layer that generates the motion estimates for the two

DoF. As in Chapters 3 and 4, the MSE is used as the loss function of the regression layer as follows (similar to equation 3.1):

$$\mathcal{L} = \frac{1}{2SW} \sum_{k=1}^{S} \|\widehat{\Delta}_k - \Delta_k\|^2, \tag{5.18}$$

where S is the number of mini-batches, k is the index that refers to the training samples in the mini-batch and W is the number of parameters to estimate, which is set to 2 for this chapter and $\widehat{\Delta} = \left[\widehat{\Delta}_x, \widehat{\Delta}_y\right]$ are the estimates in forward/backwards and sideways directions.

Layer	Kernel	Channels	Stride	Output size per channel	ReLU / Dropout
Conv1	7×4	32	2×1	400×4	Yes / Yes
Conv2	5×4	64	2×1	200×4	Yes / No
Conv3	3×4	128	2×1	100×4	Yes / No
Conv4	3×4	256	2×1	50×4	Yes / No
Conv5	3×4	512	2×1	25×4	Yes / No
Conv6	3×4	512	2×1	13×4	Yes / No
Conv7	3×4	512	2×1	7×4	Yes / No
Conv8	3×4	512	2×1	4×8	Yes / No
Conv9	3×4	512	2×1	2×4	No / No
Average	4×4	512	1×1	2×4	No / Yes

Table 5.1 Parameters of the PoseNet convolutional layers. The output size per channel is calculated considering an impulse response length of 800 taps. The output size per channel changes according to parameter configurations investigated in this work.

5.4 Experiments and results

This section presents results of numerical experiments for one transmitter and four receivers on a simulated vehicle. The transmitter is surrounded by the receivers, each 10 cm from the transmitter as shown in Fig. 5.6. The FoVs of the receivers are a narrow beam looking to the front, left, back and right of the vehicle for the receivers 1 to 4, respectively, as shown in Fig. 5.7. The use of four receivers is the most investigated case in this chapter but the use of only two receivers (forward and right looking) is also considered.

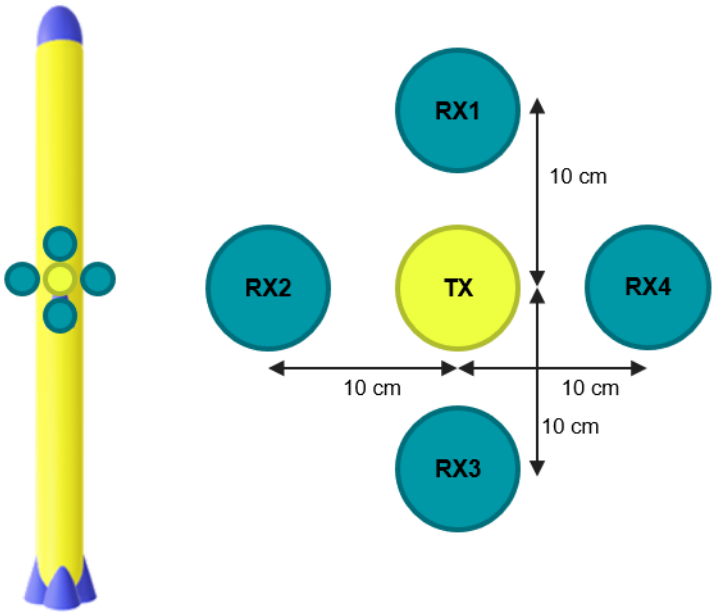


Fig. 5.6 The transmitter and receivers on the underwater vehicle.

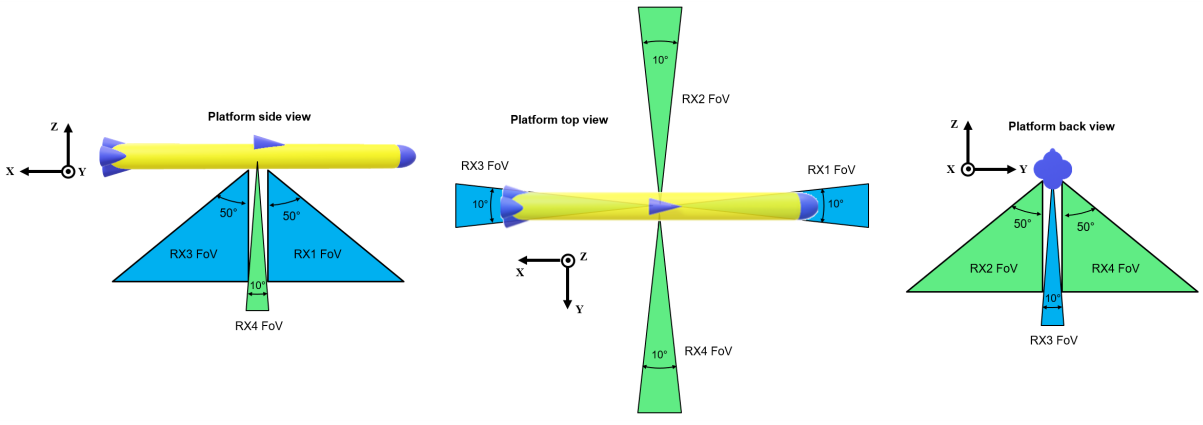


Fig. 5.7 Receivers FoV seen from the side, top and back view of the vehicle.

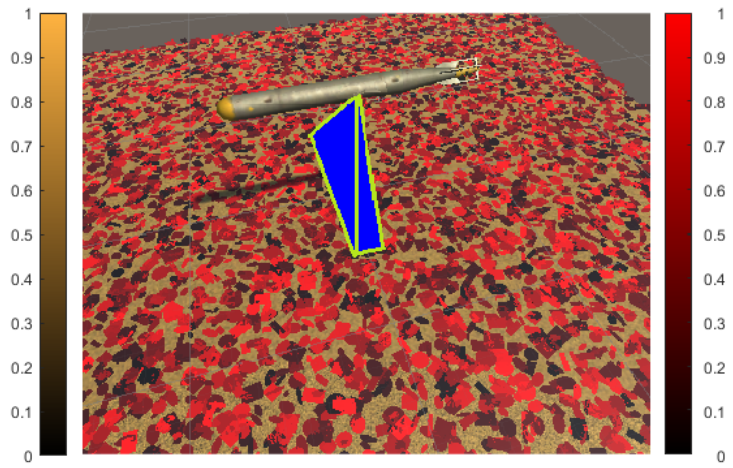


Fig. 5.8 Simulated scenario showing the underwater vehicle with the FoV of one of the receivers and the almost flat objects on the seafloor. The seafloor is divided in tiny cells whose colours are defined by the reflectivity in the range from 0 to 1, where black corresponds to 0 and the totally brown colour corresponds to 1 (Left side bar). The colours of the objects are defined by the reflectivity in the range from 0 to 1, where black corresponds to 0 and the totally red colour corresponds to 1 (Right side bar).

5.4.1 Simulated environment

The simulated environment consists of a 50 m by 50 m flat seafloor, divided into a grid of cells (2048×2048). Each cell has a reflectivity defined as a realization of random value with a uniform distribution between 0 and 1. The environment is populated by multiple objects placed on the seafloor. There are two types of objects: ellipses and rectangles. The objects are almost flat (1 mm height) while the other two dimensions (width and length) are realizations of random values uniformly distributed between 0.2 m and 1 m. The low height value is chosen to keep a simple model but later the height of the objects is increased to investigate how the motion estimates behave with taller objects. Every object has a reflectivity, which is a random value with a uniform distribution between 0 and 1. Twenty different simulated scenarios are generated to have a high variety of impulse responses for training and validation of DL networks. The difference between the scenarios is in the number and position of the objects on the seafloor. The number of objects and their position are randomly generated for each scenario. In each scenario there are between 2000 and 5000 rectangles and between

2000 and 5000 ellipses. An example of a simulated scenario with the underwater vehicle and the FoV of one of the receivers is shown in Fig. 5.8.

5.4.2 Transmitter / receivers configurations

Two DoF are considered, corresponding to the forward/backward and sideways motions of the vehicle at a constant height from the seafloor of 10 m. The vehicle velocity is a random value with a uniform probability distribution in the range from -2 m/s to 2 m/s for each DoF and a time interval between consecutive impulse response estimates (τ_r) is set to 50 ms. Therefore, the maximum speed in a particular direction is approximately 2.8 m/s. The value τ_r approximately matches to the ping period in a model of a DIDSON 300 sonar, which has been used for investigation of the motion estimation in Chapters 3 and 4.

Data sets of impulse responses are generated for various parameter sets (configurations) of the transmitter and receivers. The configurations differ in the signal parameters and the FoV size and shape. The investigated configurations are the following:

- **Two receivers:** This configuration uses two (M = 2) receivers (forward and side-looking) and Q = 2. The DL network is modified to have only two series of convolutional layers shown in Fig. 5.5b. This is the only configuration with two receivers, in the other configurations, four (M = 4) receivers are used.

- **Standard configuration:** This is considered as the basis configuration that uses the data from four (M = 4) receivers and Q = 2. The other configurations are obtained by changing one or a few parameters of this configuration. The impulse responses are generated by considering a FoV of 10° by 50° for the receivers, a bandwidth of 4.8 kHz, and a carrier frequency of 80 kHz.

- **Narrow FoVs:** This configuration uses a narrow beam of 2° by 50° in each receiver. The same frequency parameters as in the standard configuration are used.

- **Other carrier frequencies:** Four data sets are generated with the impulse responses by changing the carrier frequency f_c to 40 kHz, 60 kHz, 100 kHz and 160 kHz. The bandwidth is 4.8 kHz.

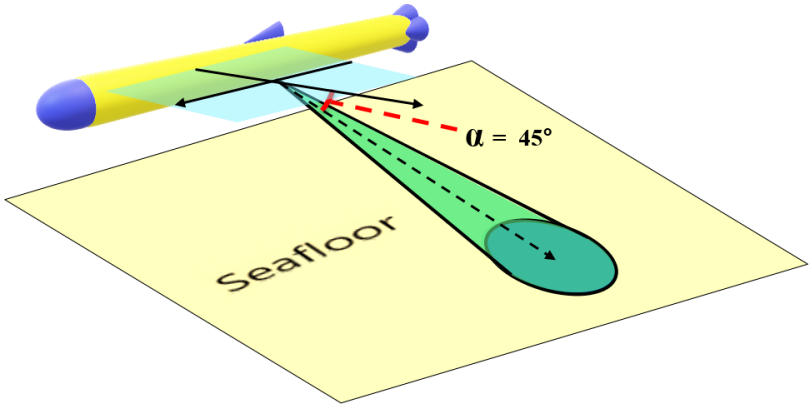


Fig. 5.9 Receiver FoV of 10° by 10° with circular shape and pitch angle of 45° producing an oval shape footprint on the seafloor.

- **Higher frequency bandwidth:** The bandwidth is set to 9.6 kHz and 19.2 kHz. The carrier frequency is 80 kHz.
- **Circular FoVs:** This configuration investigates the use of a FoV that focuses on a spot on the seafloor. It consists in a circular FoV of 10° by 10° in each receiver. It looks at a pitch angle (α) of 45° as shown in Fig. 5.9. Two data sets are generated with the bandwidth of 4.8 kHz and 19.2 kHz. The carrier frequency is 80 kHz.
- **Objects of higher height:** For this configuration, the objects on the seafloor are replaced with objects of 10 cm height rather than 1 mm to investigate how it can affect the motion estimation.
- **Groups of ten impulse responses:** The vehicle velocity is the same, but the receivers generate ten (Q = 10) impulse responses in the same time interval of 50 ms rather than only two (Q = 2) impulse responses. Groups with the ten impulse responses are packed together as a training sample. The carrier frequency is set to 80 kHz, 160 kHz or 320 kHz.

For each configuration, a data set of 3000 samples (matrices with real and imaginary parts of the impulse responses from the receivers, associated with the displacement labels) is generated and divided into training and validation sets, 80% and 20% of samples, respectively.

A DL network is trained and validated for each data set. When training the network, the Adam optimization algorithm is used [143]. The mini-batch size (S) is set to 4. The learning rate at the start of the training is set to 5×10^{-5} and it is reduced by 20% every 20 epochs. The training stops when the validation loss converges. The RMSE is used to evaluate the estimation performance for each DoF.

Table 5.2 shows results of the validation and computation time per estimate for each of the configurations. The time required per estimate is shown for a standard PC with i5-6500 CPU @3.20GHz processor without a GPU, and 8.0 GB of RAM. The number of layers of the PoseNet used for every configuration are also specified in Table 5.2. This number was observed to be the most suitable for each configuration and adding more layers showed no further improvement. For each configuration, all the convolutional layers with the exception of the last one have a ReLU activation function and batch normalization.

Bandwidth (F_d)	**Number of receivers** (M)	**Carrier frequency** (f_c)	**FoV**	**PoseNet layers**	**RMSE**		**Time per estimate**
					Fwd/bwd ***(mm)***	**Sideways** ***(mm)***	***(ms)***
4.8 kHz	2	80 kHz	$10° \times 50°$	7	5.9	6.0	7.8
	4	40 kHz	$10° \times 50°$	7	10.5	8.8	13.8
		60 kHz	$10° \times 50°$	7	6.2	6.3	38.0
		80 kHz	$10° \times 50°$	7	4.2	5.1	45.1
			$2° \times 50°$	7	2.3	2.7	44.1
			Circular $10° \times 10°$	7	15.2	11.0	8.4
		100 kHz	$10° \times 50°$	7	7.6	7.7	22.6
		160 kHz	$10° \times 50°$	8	8.84	9.05	11.9
9.6 kHz	4	80 kHz	$10° \times 50°$	8	3.8	3.9	116.6
19.2 kHz	4	80 kHz	$10° \times 50°$	9	2.4	1.9	237.5
			Circular $10° \times 10°$	9	3.5	2.0	389.4
Seafloor with up to 10 cm height objects							
4.8 kHz	4	80 kHz	$10° \times 50°$	7	5.2	5.4	49.8
Groups of 10 impulse responses							
4.8 kHz	4	80 kHz	$10° \times 50°$	6	1.7	1.8	18.1
4.8 kHz	4	160 kHz	$10° \times 50°$	6	1.51	1.58	18.0
4.8 kHz	4	320 kHz	$10° \times 50°$	7	1.27	1.53	20.4

Table 5.2 Validation errors of DL networks trained for different configurations.

From the results in Table 5.2 the following observations are made:

- The best estimation accuracy is achieved when using the groups of ten (Q = 10) impulse responses. This is due to the highest amount of information given to the network for the estimation. The greatest constraint of training a network with this technique is the time to generate the training sets, since it requires 5 times more impulse responses per training sample compared to the standard configuration. The time per estimate is comparable to most of the other configurations with the same frequency bandwidth.

- The increase of the frequency bandwidth to 9.6 kHz and further to 19.2 kHz significantly improves the estimation accuracy. The impulse responses with higher bandwidth contain more information, which improves the estimates. However, this results in a longer time interval to compute the estimate.

- The increase in the number of receivers from two to four improves the motion estimation accuracy, but even with only two receivers accurate estimates can be obtained. This is an important option if the vehicle is small, and therefore it would not be possible to accommodate many receivers.

- The narrow FoV of $2^\circ \times 50^\circ$ provides better estimates than the wider FoV of $10^\circ \times 50^\circ$.

- With the frequency bandwidth 4.8 kHz, the circular FoV of $10^\circ \times 10^\circ$ shows an increase in the RMSE compared to using the FoV of $10^\circ \times 50^\circ$. This is due to a significant reduction in the elevation angle, i.e., a smaller part of the seafloor is involved in the motion estimation. However, with the frequency bandwidth of 19.2 kHz, the reduction in the FoV less affects the performance compared to the frequency bandwidth of 4.8 kHz.

- In the case of using pairs of impulse responses, the different values of the carrier frequency show the best estimation accuracy for frequencies nearer to 80 kHz. The network parameters are modified to improve the performance for the other carrier frequency values. However, the estimation error obtained is not as low as the estimation error for 80 kHz.

- In the case of using groups of 10 impulse responses, the best estimation accuracy is for higher carrier frequencies.
- The use of higher carrier frequencies and higher bandwidth require more layers in the PoseNet to improve the estimation accuracy compared to lower frequencies. However, from the point of view of implementation, higher frequencies allow the equipment to be smaller compared to the use of lower frequencies.
- The use of 10 cm height objects on the seafloor does not affect the estimation performance much compared to the case of 1 mm height.
- In general, the motion estimation in forward/backward and sideways directions show a similar accuracy. This is because of the symmetry between both types of motion.
- It can be seen that the lower the bandwidth, the lower is the computation time, which is due to the size of the network. For a higher bandwidth the network is required to be more complex and have more layers and internal parameters. For most of the configurations, the computation time per estimate is lower than 50 ms. This makes them suitable for real-time applications considering 50 ms as the measurement interval.

5.4.3 Comparison with other techniques for motion estimation

The accuracy of the technique presented in this chapter is compared with three techniques for motion estimation. The techniques from Chapters 3 and 4 (motion estimation using consecutive sonar images as the input of the DL network and segmentation of the sonar FoV combined with compression of the images before applying them to the DL network, respectively). The third technique is the deterministic algorithm for attitude and trajectory estimation [79] previously used in Chapter 3 for an accuracy and computation speed comparison. The three techniques can estimate the motion of a vehicle in 3 DoF (forward/backward and sideways motion and yaw rotation). However, for the comparison only the motion in forward/backward and sideways is considered.

For the comparison, two proposed configurations with the best estimation performance cases are used: (i) pairs of impulse responses with F_d of 19.2 kHz, f_c of 80 kHz and FoV of $10^\circ \times 50^\circ$ and (ii) groups of ten impulse responses with F_d of 4.8 kHz, f_c of 80 kHz and FoV of $10^\circ \times 50^\circ$. The RMSE for motion estimation of all the techniques are shown in Table 5.3. It can be seen that the proposed technique significantly improves the estimation performance compared to the other techniques. Furthermore, the proposed technique could reduce the implementation cost, given that it only requires the use of one transducer and a few hydrophones, rather than an imaging sonar.

Technique	**RMSE**		**Maximum displacement** *(mm)*	**RMSE / maximum displacement**	
	Fwd/bwd *(mm)*	**Sideways** *(mm)*		**Fwd/bwd** *(%)*	**Sideways** *(%)*
[79]	8.39	1.29	20	42	6.5
Chapter 3	2.70	0.87	20	13.5	4.4
Chapter 4	0.80	1.20	20	4.0	6.0
Impulse responses method (bandwidth of 19.2 kHz)	2.37	1.90	100	2.4	1.9
Impulse responses method (groups of ten)	1.70	1.80	100	1.7	1.8

Table 5.3 RMSE for motion estimation in forward/backward and sideways directions for known and proposed techniques.

5.4.4 Comparison with other machine learning methods

The impulse responses technique presented in this chapter is compared with different machine learning techniques for regression. The MATLAB application called "Regression Learner" [158] was used to apply the techniques. A brief description of the techniques is shown in Table 5.4. For the comparison, the standard configuration data set described in subsection 5.4.2 is selected (M = 4, F_d = 4.8 kHz, f_c = 80 kHz, and FoV of 10° by 50°). Only motion in forward direction is estimated by the machine learning techniques. It is assumed that the estimation in sideways direction achieves similar results due to the similarity of both types of motions when the data set was generated.

Machine Learning technique	Description
Linear Regression	Standard Linear Regression
Fine Tree	Regression tree with minimum leaf size set to 4
Medium Tree	Regression tree with minimum leaf size set to 12
Coarse Tree	Regression tree with minimum leaf size set to 36
Linear SVM	Support Vector Machine with linear kernel function
Quadratic SVM	Support Vector Machine with quadratic kernel function
Cubic SVM	Support Vector Machine with cubic kernel function
Ensemble (Boosted trees)	Boosted trees with minimum leaf size set to 8 and number of learners set to 30
Ensemble (Bagged trees)	Bagged trees with minimum leaf size set to 8 and number of learners set to 30
Rational Quadratic GPR	Rational Quadratic Gaussian Process Regression

Table 5.4 Brief description and implementation parameters of each machine learning technique.

Table 5.5 presents results of applying the machine learning techniques for motion estimation of an underwater platform. The result of using the proposed DL network technique is also included. It can be seen that among the machine learning techniques, the best estimation is achieved with the Ensemble (Bagged trees) technique with an RMSE of 17.77 mm. However, the accuracy is much better for the DL network technique presented in this chapter, achieving an RMSE of 4.2 mm.

5.4.5 Trajectory reconstruction using simulated data

The DL network trained with the data set of 19.2 kHz is used to estimate the displacements in simulated trajectories and recover the trajectories. The impulse responses obtained in the trajectories are grouped into pairs of consecutive positions of the vehicle. Each pair of impulse responses is given to the trained DL network to estimate the displacement between these two positions. The vehicle trajectory is recovered using these displacement estimates.

The points of a trajectory are represented as x_i and y_i, where i is an index that refers to the position number in the trajectory. The initial position x_1, y_1 of the estimated trajectory is set to zero. Since only 2 DoF are considered for the experiments in this chapter, equation3.4 is

Machine Learning technique	**RMSE *(mm)***
Linear Regression	58.01
Fine Tree	31.71
Medium Tree	30.69
Coarse Tree	32.28
Linear SVM	53.59
Quadratic SVM	29.07
Cubic SVM	28.45
Ensemble (Boosted trees)	23.33
Ensemble (Bagged trees)	17.77
Rational Quadratic GPR	23.81
DL network	4.2

Table 5.5 Validation errors for each machine learning technique.

modified to obtain the other points of the trajectory are calculated by adding the displacement estimate in each DoF:

$$\begin{aligned} x_{i+1} &= x_i + \widehat{\Delta}_{x_i}, \\ y_{i+1} &= y_i + \widehat{\Delta}_{y_i}, \end{aligned} \tag{5.19}$$

where $\widehat{\Delta}_{x_i}$ and $\widehat{\Delta}_{y_i}$ are displacement estimates in the forward/backward and sideways directions, respectively.

Six trajectories are generated in the simulator as shown in Fig. 5.10. The description of each trajectory is as follows:

- Trajectory 1 (Fig. 5.10a): It has a square shape with starting and finishing points at the same place (loop trajectory). The vehicle moves in all four directions at different time intervals (forward, right, backward, left). The size of each side of the square is 5 m. The trajectory is built with 400 positions (399 displacements). The first motion (left) consists of 100 displacements at a speed of 1 m/s, the second motion (forward) has 100 displacements at a speed of 1 m/s, the third motion (right) consists of 50 displacements at a speed of 1.5 m/s and 50 displacements at a speed of 0.5 m/s, the last

motion (backwards) has 50 displacements at a speed of 1.5 m/s and 50 displacements at a speed of 0.5 m/s.

- Trajectory 2 (Fig. 5.10b): It has forward and right motion in a stair shape. The trajectory consists of 400 positions with different speeds the same as in trajectory 1 but in different directions.

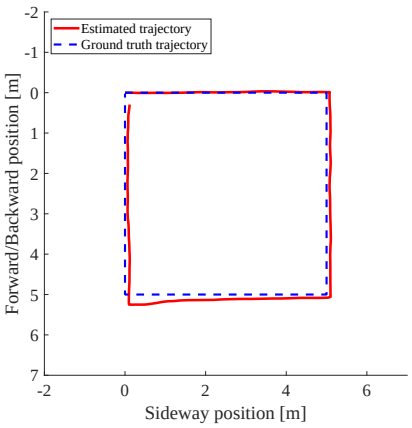
- Trajectory 3 (Fig. 5.10c): The vehicle moves in a zigzag shape. The trajectory consists of 340 positions. The sequence of displacements consists of 80 displacements to the backward at a speed of 1 m/s, 50 left displacements at 1 m/s, 40 forward displacements at 0.5 m/s, 50 left displacements at 1.5 m/s, 40 backward displacements at 0.5 m/s, 50 left displacements at 0.5 m/s, and 30 forward displacements at 1.5 m/s.

- Trajectory 4 (Fig. 5.10d): It has motion in all four directions to make a circle shape. The radius of the circle is 5 m. The trajectory is made of 400 positions while the vehicle moves at a speed of approximately 1.5 m/s.

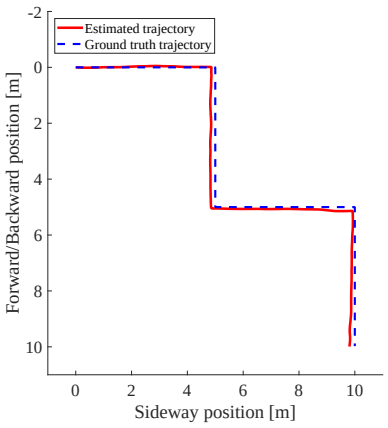
- Trajectory 5 (Fig. 5.10e): It is of a sinusoid shape combining forward motion while moving sideways. The trajectory consists of 400 positions. The vehicle changes the speed across the trajectory, with maximum speed of approximately 2 m/s and minimum speed of approximately 0.6 m/s.

- Trajectory 6 (Fig. 5.10f): It has motions in forward and right directions at the same time to make a combined diagonal motion. The trajectory is made of 200 positions at a constant speed of approximately 1.4 m/s (1 m/s forward speed and 1 m/s left motion speed).

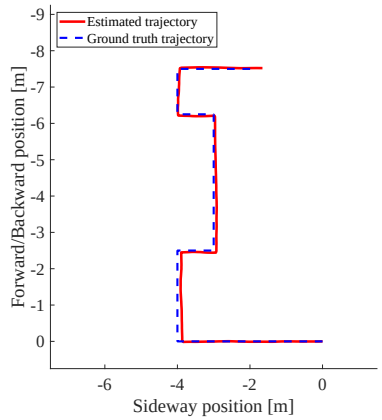
Comparing the ground truth and estimated trajectories, it can be seen that the estimated trajectories accurately follow the ground truth with a small error. The RMSE for each of the estimated trajectories in two DoF can be seen in Table 5.6 as well as the final error magnitude (difference between the point where the estimated trajectory should end and the point where it actually does). The RMSE is in the range from 1.5 mm to 3.2 mm for all the trajectories



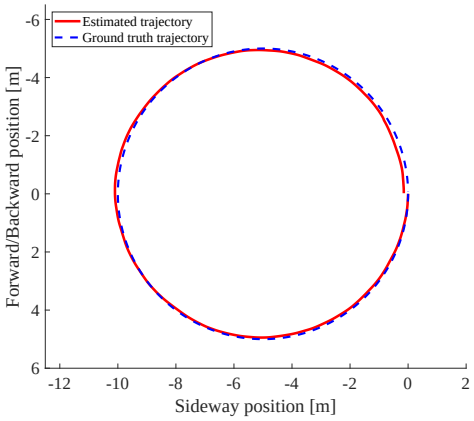
(a)



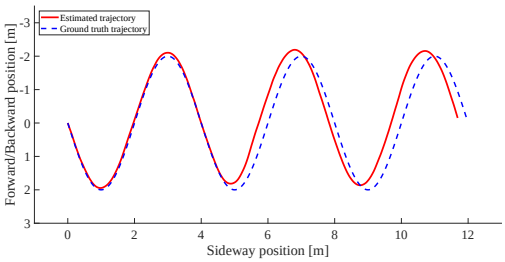
(b)



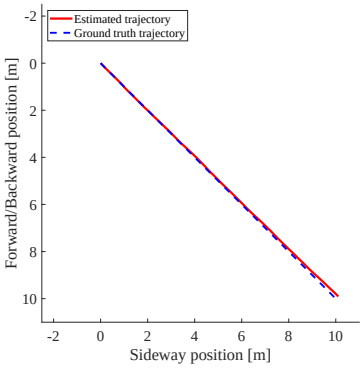
(c)



(d)



(e)



(f)

Fig. 5.10 Six simulated trajectories. Blue line is the ground truth trajectory and red line is the estimated trajectory.

and the highest error magnitude is 31 cm, which is a high accuracy considering that the total travelled distances are higher than 14 m.

The error in the loop trajectories (1 and 4) is also measured according to how far from the initial point is the ending point. For trajectory 1 (square shape), the error is 31 cm and for trajectory 4 (circle shape) the error is 15 cm. The total distance travelled by the vehicle over these two trajectories is 20 m and about 31.42 m, respectively, which makes the final error of about 1.6% and 0.5% of the total travelled distance, respectively.

Trajectory	**RMSE**		**Error magnitude** *(cm)*
	Fwd/bwd *(mm)*	**Sideways** *(mm)*	
1	2.08	3.16	31
2	3.26	2.60	19
3	3.35	2.93	10
4	2.08	3.30	15
5	2.08	3.16	19
6	2.18	1.51	14

Table 5.6 RMSE for motion estimation in forward/backward and sideways directions for every simulated trajectory and error magnitude of the point where the estimated trajectory should end and the actual value.

5.4.6 Comparison of using impulse responses versus using sonar images

An attempt was made to estimate the trajectories shown in Fig. 5.10 by using the technique proposed in Chapter 3. However the quality of most of the recovered trajectories was not good. For that reason, only the reconstruction of trajectory 5 is shown in this subsection.

To generate the equivalent estimated trajectory, the motion estimation RMSE of the sonar images method is used. The RMSE for forward/backward motion is 1.0 mm and for the sideways motion is 2.8 mm for the sonar images method. The ground truth trajectory is modified by introducing an error using the RMSE in each DoF for every pair of positions where an image (or impulse response) is obtained. The error is defined as a randomly generated value with a Gaussian distribution between ± 1.0 mm for forward/backward

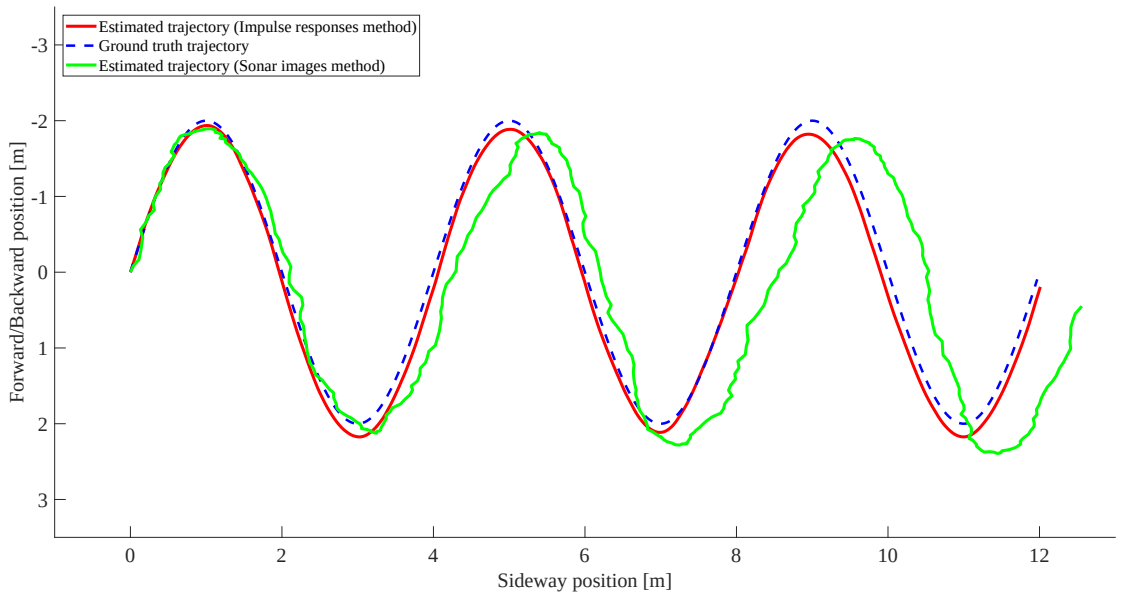


Fig. 5.11 Estimated trajectories using the impulse response method (this chapter) in red line and the method of sonar images (Chapter 3) in green line; the ground truth trajectory is shown as the dashed blue line.

estimates and ± 2.8 mm for sideways estimates. The resulting trajectory with the added error is compared with the estimated trajectory using the impulse responses method and the ground truth trajectory in Fig. 5.11. It can be seen that the trajectory estimated by the impulse responses method is smoother and more accurate than estimated by the sonar images method.

5.5 Conclusions

In this chapter, a DL method was developed for motion estimation of an underwater vehicle that uses the impulse responses obtained for (two or four) receivers mounted on the vehicle. A model to generate the impulse responses in an underwater simulator is presented and used to produce different data sets to train the DL network. The data sets differ from each other in the parameters used to generate the impulse responses such as the bandwidth and carrier frequency. When generating the impulse responses, a delay compensation technique

is applied before calculating the impulse responses, to consider the Doppler effect due to the vehicle motion.

The motion estimation techniques are validated with simulated data. The proposed technique shows a similar or higher accuracy than techniques that use sonar images rather than impulse responses. Furthermore, the use of impulse responses can reduce significantly the cost of implementation compared to the use of sonar images. When compared with other machine learning techniques for regression, the proposed method shows a much smaller RMSE. The proposed DL network achieves an RMSE 4 times smaller than the best of the considered standard machine learning techniques.

Six simulated trajectories are presented and fully estimated with the impulse responses method, showing high estimation accuracy and almost replicating the ground truth trajectory. For two loop trajectories, where the starting point and the ending point in the underwater scenario are the same, low errors of 1.6% and 0.5% are achieved.

Chapter 6

Conclusions and further work

6.1 Conclusions

The work in this thesis focuses on the solution of problems associated with underwater navigation and concerns with its accuracy and complexity. To deal with these problems, the use of acoustic signals obtained by sonar sensors or a set of simple hydrophones is considered. More specifically, this work centers on the motion estimation of a vehicle carrying the sensors. Known techniques for underwater navigation are based on the use of underwater beacons, inertial sensors, DVL, CVL, dead reckoning techniques and techniques that take reference from the environment like image registration with sonar images.

The first step in this work was the development of a sonar simulator. This was a key for the following parts of the research. Without the simulator, the implementation of DL techniques would have been a tremendous challenge. The simulator generates realistic images and it has the precision to provide the ground truth data required for training and validating the techniques for motion estimation.

The motion estimation techniques implemented in this work are based on the use of DL networks. They are fed with acoustic images or channel impulse responses and they output the motion estimates in three or two DoF. The DL networks have been trained and validated using large data sets generated by the sonar simulator. Several state-of-the-art DL architectures focused on optical images have been adapted and optimized to work with the acoustic signal.

The optimized DL architectures combined with different sensor configurations investigated in this work are able to achieve a high accuracy compared to other techniques from the literature and a reduction in the computation complexity to reduce the estimation time and make the methods feasible for real-time applications.

The contributions of the work presented in this thesis are summarized as follows:

1. A new sonar simulator has been developed and implemented, using the Unity and MATLAB design platforms. The initial version of the simulator described in Chapter 2 is capable of generating acoustic images for collocated transmit and receive sonar antennas for 3D scenarios. Its modified (improved) versions presented in further chapters are capable of generating realistic sonar images for arbitrary positioned sonar antennas, and finally channel impulse responses. The simulator allows customisation of sensor parameters such as FoV, resolution (number of rays in the FoV), maximum measured range, etc., and it can also be adjusted in accordance with real sensors. The simulated sonar images can be noiseless or with noise. The noise model can also be adjusted. Using the simulator, large data sets of ground truth data to validate motion estimation techniques can be generated.

2. A collection of data sets of simulated acoustic (sonar) images with their respective labels that represent the underwater sonar platform motion. Multiple data sets have been generated and can be used later for validation of new methods of image registration and motion estimation, regardless if it is a DL or deterministic method.

3. Several DL networks are implemented and compared to identify the most suitable for motion estimation using sonar images. The DL networks are validated using noiseless and noisy images. The results show that a millimetre accuracy for translation motion and below 0.1° for rotation motion can be achieved. The DL methods are also compared with a deterministic method, showing a much lower computation time and more accurate motion estimates than the deterministic algorithm. It was observed that networks trained with simulated data and then used for estimation with real data sets

can obtain good estimates even when the sonar features between the training set and the real data sets are different.

4. A technique that consists in converting sonar images into vectors by adding up the pixels in each row. This reduces the size of the DL networks that are used to estimate the motion. This technique showed significant reduction in the computation time of up to 10 times compared to techniques that use images.

5. The division of the FoV of a simulated sonar into four quadrants. An image is generated from each quadrant. This is combined with the vector technique by converting the images into vectors and grouped together as the input of the DL network. The FoV division approach showed a high accuracy compared to using the whole FoV or different portions of it.

6. A motion estimation method, enabled by full-duplex operation, based on DL analysis of time variation of complex-valued channel impulse responses. The full-duplex operation allows a continuous transmission and reception of signals by two or four receivers.

6.2 Further work

In this section, suggestions are given for further work based on this thesis. They are summarized as follows:

- After the successful results shown using the simulator to generate data, it could be extended to other application areas, such as the synthetic aperture sonar. The noise model applied to the sonar images can also be customized to specific sonars by measuring the type of noise present in the simulated sensors to define their own noise model.

- The results obtained for the case of compressing acoustic images into vectors suggest that further work could improve even more the motion estimation accuracy. This could

be done by proposing other preprocessing of sonar images before feeding them to a DL network or with other variants of splitting the sonar FoV.

- The DL network requires retraining with a new data set if the sonar parameters or position and number of receivers change. One solution for this can be the use of multiple pretrained networks whose training data set is adjusted to different sonar configurations.

- At this moment, the motion estimation based on the channel impulse responses technique has been investigated only for two DoF. The estimation for three DoF is considered the next problem to be investigated. Also, optimizing the configuration of the sensors such as the distance and orientation relative to the transmitter and the FoV could improve further the estimation accuracy. The results obtained so far are a hint to future work to develop a system for motion estimation using signals obtained from a few single hydrophones. Moreover, the shape of the hydrophone's beam is crucial as shown when comparing the spot and the wider FoVs and further research should be done on its optimization.

- It can be seen that even with the simple trajectory recovery procedure described in the thesis, the estimated trajectory is close to the ground truth. More sophisticated recovery techniques based on regularized spline interpolation and/or Kalman filtering could improve the trajectory estimation.

Acronyms

2D Two-dimensional

3D Three-dimensional

AUV Autonomous Underwater Vehicle

CVL Correlation Velocity Log

DL Deep Learning

DVL Doppler Velocity Log

DoF Degrees of Freedom

FLS Forward-Looking Sonar

FoV Field of View

INS Inertial Navigation System

LBL Long Baseline

MSE Mean Squared Error

RMSE Root Mean Square Error

ROS Robot Operating System

ROV Remotely Operated Vehicle

ReLU Rectified Linear Unit activation function

SBL Short Baseline

USBL Ultra Short Baseline

References

[1] F. Jalal and F. Nasir, "Underwater navigation, localization and path planning for autonomous vehicles: A review," in *IEEE International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, 2021, pp. 817–828.

[2] *DIDSON 300 Standard Version Specifications*, Sound Metrics Corp., available online: http://www.soundmetrics.com/products/DIDSON-Sonars/DIDSON-300m/DIDSON-300-Standard-Version-Specifications (accessed on 08 July 2021).

[3] L. Conti, M. Rodriques, and B. Hanot, "Hydroelectric power plant inspections," *Hydro International Magazine*, vol. 20, no. 4, pp. 16–19, 2016.

[4] "Acquest websiste," https://www.acquest.com.br, accessed: 2021-01-28.

[5] J. E. Almanza-Medina, B. T. Henson, and Y. V. Zakharov, "Imaging sonar simulator for assessment of image registration techniques," in *MTS/IEEE OCEANS Seattle*, 2019, pp. 1–7.

[6] J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, "Deep learning architectures for navigation using forward looking sonar images," *IEEE Access*, vol. 9, pp. 33 880–33 896, 2021.

[7] J. E. Almanza-Medina, B. Henson, and Y. Zakharov, "Motion estimation of an underwater platform using images from two sonar sensors," *TechRxiv preprint techrxiv.15059991.v1*, 2021.

[8] J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov, "Sonar FoV segmentation for motion estimation using DL networks," *IEEE Access*, vol. 10, pp. 25 591–25 604, 2022.

[9] J. E. Almanza-Medina, B. Henson, L. Shen, and Y. V. Zakharov, "Motion estimation of underwater platforms using impulse responses from the seafloor," *IEEE Access*, vol. 10, pp. 127 047–127 060, 2022.

[10] C.-C. Kao, Y.-S. Lin, G.-D. Wu, and C.-J. Huang, "A comprehensive study on the internet of underwater things: Applications, challenges, and channel models," *Sensors*, vol. 17, no. 7, p. 1477, 2017.

[11] M. M. Dos Santos, G. G. De Giacomo, P. L. Drews, and S. S. Botelho, "Underwater sonar and aerial images data fusion for robot localization," in *19th IEEE International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil*, 2019, pp. 578–583.

[12] Z. Jin, Q. Zhao, and Y. Luo, "Routing void prediction and repairing in AUV-assisted underwater acoustic sensor networks," *IEEE Access*, vol. 8, pp. 54 200–54 212, 2020.

[13] R. Nwosu and J. Peter, "A review on underwater acoustic sensor network," *Australian Journal of Science and Technology*, vol. 5, no. 4, pp. 678–685, 2021.

[14] E. Zereik, M. Bibuli, N. Mišković, P. Ridao, and A. Pascoal, "Challenges and future trends in marine robotics," *Annual Reviews in Control*, vol. 46, pp. 350–368, 2018.

[15] H. Yu, W. Lu, D. Liu, Y. Han, and Q. Wu, "Speeding up gaussian belief space planning for underwater robots through a covariance upper bound," *IEEE Access*, vol. 7, pp. 121 961–121 974, 2019.

[16] H. Huang, J. Tang, B. Zhang, J. Chen, J. Zhang, and X. Song, "A novel nonlinear algorithm for non-gaussian noises and measurement information loss in underwater navigation," *IEEE Access*, vol. 8, pp. 118 472–118 484, 2020.

[17] A. Grządziel, "Results from developments in the use of a scanning sonar to support diving operations from a rescue ship," *Remote Sensing*, vol. 12, no. 4, p. 693, 2020.

[18] M. Specht, A. Stateczny, C. Specht, S. Widźgowski, O. Lewicka, and M. Wiśniewska, "Concept of an innovative autonomous unmanned system for bathymetric monitoring of shallow waterbodies (innobat system)," *Energies*, vol. 14, no. 17, p. 5370, 2021.

[19] L. Whitcomb, D. R. Yoerger, H. Singh, and J. Howland, "Advances in underwater robot vehicles for deep ocean exploration: navigation, control, and survey operations," in *Robotics Research*, J. M. Hollerbach and D. E. Koditschek, Eds. London: Springer, 2000, pp. 439–448.

[20] M. B. Loc, H.-S. Choi, J.-M. Seo, S.-H. Baek, and J.-Y. Kim, "Development and control of a new auv platform," *International Journal of Control, Automation and Systems*, vol. 12, no. 4, pp. 886–894, 2014.

[21] M. Salhaoui, J. C. Molina-Molina, A. Guerrero-González, M. Arioua, and F. J. Ortiz, "Autonomous underwater monitoring system for detecting life on the seabed by means of computer vision cloud services," *Remote Sensing*, vol. 12, no. 12, p. 1981, 2020.

[22] X. Yuan, J.-F. Martínez-Ortega, J. A. S. Fernández, and M. Eckert, "Aekf-slam: A new algorithm for robotic underwater navigation," *Sensors*, vol. 17, no. 5, p. 1174, 2017.

[23] V. Bobkov, A. Kudryashov, and A. Inzartsev, "A technique to navigate autonomous underwater vehicles using a virtual coordinate reference network during inspection of industrial subsea structures," *Remote Sensing*, vol. 14, no. 20, p. 5123, 2022.

[24] N. Modalavalasa, G. S. Rao, and K. S. Prasad, "An efficient implementation of tracking using kalman filter for underwater robot application," *International Journal of Computer Science, Engineering and Information Technology*, vol. 2, no. 2, pp. 67–78, 2012.

[25] C. Meek and Z. Song, "Fast autonomous underwater exploration using a hybrid focus model with semantic representation," in *OCEANS MTS/IEEE Seattle*, 2019, pp. 1–6.

[26] I. Klein and R. Diamant, "Dead reckoning for trajectory estimation of underwater drifters under water currents," *Journal of Marine Science and Engineering*, vol. 8, no. 3, p. 205, 2020.

[27] J. González-García, A. Gómez-Espinosa, E. Cuan-Urquizo, L. G. García-Valdovinos, T. Salgado-Jiménez, and J. A. E. Cabello, "Autonomous underwater vehicles: Localization, navigation, and communication for collaborative missions," *Applied Sciences*, vol. 10, no. 4, p. 1256, 2020.

[28] R. Garcia and N. Gracias, "Detection of interest points in turbid underwater images," in *IEEE OCEANS 2011, Santander, Spain*, 2011, pp. 1–9.

[29] N. Hurtós, D. Ribas, X. Cufí, Y. Petillot, and J. Salvi, "Fourier-based registration for robust forward-looking sonar mosaicing in low-visibility underwater environments," *Journal of Field Robotics*, vol. 32, no. 1, pp. 123–151, 2015.

[30] F. Ferreira, D. Machado, G. Ferri, S. Dugelay, and J. Potter, "Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art," in *OCEANS MTS/IEEE, Monterey, USA*, 2016, pp. 1–6.

[31] D. P. Williams and S. Dugelay, "Multi-view SAS image classification using deep learning," in *MTS/IEEE OCEANS, Monterey*, 2016, pp. 1–9.

[32] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional neural network with data augmentation for SAR target recognition," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 3, pp. 364–368, 2016.

[33] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," in *56th IEEE Annual Conference on Decision and Control (CDC)*, 2017, pp. 2724–2731.

[34] S. Lee, "Deep learning of submerged body images from 2D sonar sensor based on convolutional neural network," in *Proceedings of the IEEE Underwater Technology (UT) Conference, Busan, South Korea*, 2017, pp. 1–3.

[35] X. Wang, J. Jiao, J. Yin, W. Zhao, X. Han, and B. Sun, "Underwater sonar image classification using adaptive weights convolutional neural network," *Applied Acoustics*, vol. 146, pp. 145–154, 2019.

[36] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[38] K. Fang, S. Zhang, and J. Liu, "Principle and error analysis of doppler acoustic omni-directional beacon," in *Geo-Informatics in Resource Management and Sustainable Ecosystem, Wuhan, China, November 8-10*. Berlin, Heidelberg: Springer, 2013, pp. 384–392.

[39] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-graph slam using forward-looking sonar," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2330–2337, 2018.

[40] Ø. Sture, P. Norgren, and M. Ludvigsen, "Trajectory planning for navigation aiding of autonomous underwater vehicles," *IEEE Access*, vol. 8, pp. 116 586–116 604, 2020.

[41] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2013.

[42] J. Zhang, Y. Han, C. Zheng, and D. Sun, "Underwater target localization using long baseline positioning system," *Applied Acoustics*, vol. 111, pp. 129–134, 2016.

[43] Y. Xu, W. Liu, X. Ding, P. Lv, C. Feng, B. He, and T. Yan, "USBL positioning system based adaptive Kalman filter in AUV," in *MTS/IEEE OCEANS Kobe Techno-Oceans (OTO)*, 2018, pp. 1–4.

[44] Y. Yang, Y. Xiao, and T. Li, "A survey of autonomous underwater vehicle formation: performance, formation control, and communication capability," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 815–841, 2021.

[45] S. Fan, C. Liu, B. Li, Y. Xu, and W. Xu, "Auv docking based on usbl navigation and vision guidance," *Journal of Marine Science and Technology*, vol. 24, no. 3, pp. 673–685, 2019.

[46] F. Dubrovin, Y. Vaulin, A. Scherbatyuk, D. Scherbatyuk, and A. Rodionov, "Navigation for auv, located in the shadow area of lbl, during the group operations," in *IEEE Global Oceans, Singapore–US Gulf Coast*, 2020, pp. 1–6.

[47] W. Chen and R. Sun, "Optimal distance for moving long baseline positioning system with distance-dependent measurement noise," *Advances in Mechanical Engineering*, vol. 10, no. 6, pp. 1–9, 2018.

[48] Y. Cong, C. Gu, T. Zhang, and Y. Gao, "Underwater robot sensing technology: A survey," *Fundamental Research*, vol. 1, no. 3, pp. 337–345, 2021.

[49] W. Yan, W. Chen, and R. Cui, "Moving long baseline positioning algorithm with uncertain sound speed," *Journal of Mechanical Science and Technology*, vol. 29, no. 9, pp. 3995–4002, 2015.

[50] D. Pick, E. Wolbrecht, M. Anderson, D. Edwards, and J. Canning, "Uncertainty analysis of ultra-short-and long-baseline localization systems for autonomous underwater vehicles," in *MTS/IEEE OCEANS Charleston*, 2018, pp. 1–6.

[51] L. N. P. Pasnani and M. L. Seto, "Underwater gravity-based navigation for long-range AUV missions," *Maritime Engineering Journal*, no. 94, pp. 18–21, 2020.

[52] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 33, no. 7, pp. 994–1012, 2016.

[53] M. V. Jakuba, J. C. Kinsey, J. W. Partan, and S. E. Webster, "Feasibility of low-power one-way travel-time inverted ultra-short baseline navigation," in *MTS/IEEE OCEANS, Washington, USA*, 2015, pp. 1–10.

[54] A. Grosch, C. Enneking, L. A. Greda, D. Tanajewski, G. Grunwald, and A. Ciećko, "Theoretical concept for a mobile underwater radio-navigation system using pseudolite buoys," *Remote Sensing*, vol. 12, no. 21, p. 3636, 2020.

[55] P. N. Mikhalevsky, B. J. Sperry, K. F. Woolfe, M. A. Dzieciuch, and P. F. Worcester, "Deep ocean long range underwater navigation," *The Journal of the Acoustical Society of America*, vol. 147, no. 4, pp. 2365–2382, 2020.

[56] H. Zhao, Z. Wang, S. Qiu, Y. Shen, L. Zhang, K. Tang, and G. Fortino, "Heading drift reduction for foot-mounted inertial navigation system via multi-sensor fusion and dual-gait analysis," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8514–8521, 2018.

[57] W. Gao, Y. Zhang, and J. Wang, "Research on initial alignment and self-calibration of rotary strapdown inertial navigation systems," *Sensors*, vol. 15, no. 2, pp. 3154–3171, 2015.

[58] Q. Wu, K. Li, and J. Liu, "The asynchronous gimbal-rotation-based calibration method for lever-arm errors of two rotational inertial navigation systems," *IEEE Access*, vol. 7, pp. 4653–4663, 2018.

[59] A. Karmozdi, M. Hashemi, H. Salarieh, and A. Alasty, "Ins-dvl navigation improvement using rotational motion dynamic model of auv," *IEEE Sensors Journal*, vol. 20, no. 23, pp. 14 329–14 336, 2020.

[60] A. Tal, I. Klein, and R. Katz, "Inertial navigation system/doppler velocity log (ins/dvl) fusion with partial dvl measurements," *Sensors*, vol. 17, no. 2, p. 415, 2017.

[61] Z. Song and K. Mohseni, "Long-term inertial navigation aided by dynamics of flow field features," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 4, pp. 940–954, 2017.

[62] P. Boltryk, M. Hill, A. Keary, B. Phillips, H. Robinson, and P. White, "An ultrasonic transducer array for velocity measurement in underwater vehicles," *Ultrasonics*, vol. 42, no. 1-9, pp. 473–478, 2004.

[63] D. Li, J. Xu, H. He, and M. Wu, "An underwater integrated navigation algorithm to deal with dvl malfunctions based on deep learning," *IEEE Access*, vol. 9, pp. 82 010–82 020, 2021.

[64] S. Wirth, P. L. N. Carrasco, and G. O. Codina, "Visual odometry for autonomous underwater vehicles," in *MTS/IEEE OCEANS, Bergen*, 2013, pp. 1–6.

[65] T. E. Blanford, D. C. Brown, and R. J. Meyer Jr, "Design considerations for a compact correlation velocity log," in *Proceedings of Meetings on Acoustics 175ASA*, vol. 33, no. 1. Acoustical Society of America, 2018, p. 070003.

[66] I. B. Saksvik, A. Alcocer, and V. Hassani, "A deep learning approach to dead-reckoning navigation for autonomous underwater vehicles with limited sensor payloads," in *OCEANS, San Diego–Porto*, 2021, pp. 1–9.

[67] E. Topini, A. Topini, M. Franchi, A. Bucci, N. Secciani, A. Ridolfi, and B. Allotta, "LSTM-based dead reckoning navigation for autonomous underwater vehicles," in *IEEE Global Oceans, Singapore–US Gulf Coast*, 2020, pp. 1–7.

[68] M. T. Sabet, H. M. Daniali, A. Fathi, and E. Alizadeh, "A low-cost dead reckoning navigation system for an auv using a robust ahrs: Design and experimental analysis," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 4, pp. 927–939, 2017.

[69] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[70] M. Aladem and S. A. Rawashdeh, "Lightweight visual odometry for autonomous mobile robots," *Sensors*, vol. 18, no. 9, p. 2837, 2018.

[71] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip *et al.*, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[72] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, 2018.

[73] S. S. da Costa Botelho, P. Drews, G. L. Oliveira, and M. da Silva Figueiredo, "Visual odometry and mapping for underwater autonomous vehicles," in *6th IEEE Latin American Robotics Symposium (LARS), 2009*, 2009, pp. 1–6.

[74] A. Kim and R. M. Eustice, "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 719–733, 2013.

[75] M. Meireles, R. Lourenço, A. Dias, J. M. Almeida, H. Silva, and A. Martins, "Real time visual SLAM for underwater robotic inspection," in *MTS/IEEE OCEANS 2014, St. John's, Newfoundland*, 2014, pp. 1–5.

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

[77] Z. Xu, M. Haroutunian, A. J. Murphy, J. Neasham, and R. Norman, "An integrated visual odometry system for underwater vehicles," *IEEE Journal of Oceanic Engineering*, vol. 46, no. 3, pp. 848–863, 2020.

[78] H. Assalih, S. Negahdaripour, and Y. Petillot, "3-D motion estimation in passive navigation by acoustic imaging," in *MTS/IEEE OCEANS Seattle*, 2010, pp. 1–6.

[79] B. T. Henson and Y. V. Zakharov, "Attitude-trajectory estimation for forward-looking multibeam sonar based on acoustic image registration," *IEEE Journal of Oceanic Engineering*, vol. 44, no. 3, pp. 753–766, 2018.

[80] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[81] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[82] A. Stateczny, W. Błaszczak-Bąk, A. Sobieraj-Żłobińska, W. Motyl, and M. Wisniewska, "Methodology for processing of 3d multibeam sonar big data for comparative navigation," *Remote Sensing*, vol. 11, no. 19, p. 2245, 2019.

[83] M. Franchi, A. Ridolfi, and B. Allotta, "Underwater navigation with 2d forward looking sonar: An adaptive unscented kalman filter-based strategy for auvs," *Journal of Field Robotics*, vol. 38, no. 3, pp. 355–385, 2021.

[84] A. Karmozdi, M. Hashemi, H. Salarieh, and A. Alasty, "Implementation of translational motion dynamics for ins data fusion in dvl outage in underwater navigation," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 6652–6659, 2020.

[85] F. Sun, W. Xu, and J. Li, "Enhancement of the aided inertial navigation system for an auv via micronavigation," in *MTS/IEEE OCEANS, Seattle*, 2010, pp. 1–4.

[86] S. Caporale and Y. Petillot, "A new framework for synthetic aperture sonar micronavigation," *arXiv preprint arXiv:1707.08488*, 2017.

[87] Y. Yang and G. Huang, "Acoustic-inertial underwater navigation," in *International Conference on Robotics and Automation (ICRA), Singapore*, 2017, pp. 4927–4933.

[88] N. Y. Ko, D. Jeong, and G. Song, "Navigation of a remotely operated underwater vehicle using IMU and DVL," in *19th IEEE International Conference on Control, Automation and Systems (ICCAS), Jeju, South Korea*, 2019, pp. 1098–1100.

[89] P. A. Miller, J. A. Farrell, Y. Zhao, and V. Djapic, "Autonomous underwater vehicle navigation," *IEEE Journal of Oceanic Engineering*, vol. 35, no. 3, pp. 663–678, 2010.

[90] Y. Wang, X. Ma, J. Wang, and H. Wang, "Pseudo-3d vision-inertia based underwater self-localization for auvs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7895–7907, 2020.

[91] H. Saç, M. K. Leblebicioğlu, and G. Bozdağı Akar, "2D high-frequency forward-looking sonar simulator based on continuous surfaces approach," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, no. 1, pp. 2289–2303, 2015.

[92] T. Guerneve, K. Subr, and Y. Petillot, "Three-dimensional reconstruction of underwater objects using wide-aperture imaging sonar," *Journal of Field Robotics*, vol. 35, no. 6, pp. 890–905, 2018.

[93] M. Prats, J. Perez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *IEEE/RSJ international conference on Intelligent Robots and Systems*, 2012, pp. 2577–2582.

[94] T. Guerneve and Y. Petillot, "Underwater 3D reconstruction using BlueView imaging sonar," in *IEEE OCEANS*, 2015, pp. 1–7, Genova, Italy.

[95] H. Ragheb and E. R. Hancock, "Lambertian reflectance correction for rough and shiny surfaces," in *IEEE Proceedings. International Conference on Image Processing, Rochester, NY, USA*, vol. 2, 2002, pp. 553–556.

[96] M. Oren and S. K. Nayar, "Generalization of Lambert's reflectance model," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, Orlando, Florida, USA*, 1994, pp. 239–246.

[97] *P900 Series 2D Imaging Sonar*, BlueView Technologies, Inc., 2011, rev. 1.

[98] A. S. Glassner, *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[99] J. M. Bell, "Application of optical ray tracking techniques to the simulation of sonar images," *Optical Engineering*, vol. 36, no. 6, pp. 1806–1813, 1997.

[100] C. Morency, D. J. Stilwell, and S. Hess, "Development of a simulation environment for evaluation of a forward looking sonar system for small AUVs," in *MTS/IEEE OCEANS Seattle*, 2019, pp. 1–9.

[101] T. Watanabe, G. Neves, R. Cerqueira, T. Trocoli, M. Reis, S. Joyeux, and J. Albiez, "The rock-gazebo integration and a real-time auv simulation," in *12th IEEE Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 2015, pp. 132–138.

[102] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez, and L. Oliveira, "A novel GPU-based sonar simulator for real-time applications," *Computers & Graphics*, vol. 68, pp. 66–76, 2017.

[103] M. Borawski and P. Forczmański, "Sonar image simulation by means of ray tracing and image processing," in *Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems*. Springer, 2005, pp. 209–214.

[104] M. D. Aykin and S. Negahdaripour, "Efficient ray-casting of quadric surfaces for forward-scan sonars," in *MTS/IEEE OCEANS, Monterey*, 2016, pp. 1–7, Monterey.

[105] D. Gueriot and C. Sintes, "Sonar data simulation," in *Sonar Systems*. IntechOpen, 2011.

[106] "Unity," Available online: https://unity.com/ (accessed on 08 July 2022).

[107] S. L. Kim, H. J. Suk, J. H. Kang, J. M. Jung, T. H. Laine, and J. Westlin, "Using unity 3D to facilitate mobile augmented reality game development," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 21–26.

[108] J. Ribeiro, J. E. Almeida, R. J. Rossetti, A. Coelho, and A. L. Coelho, "Using serious games to train evacuation behaviour," in *The 7th IEEE Iberian Conference on Information Systems and Technologies (CISTI 2012)*, 2012, pp. 1–6.

[109] A. Indraprastha and M. Shinozaki, "The investigation on using Unity3D game engine in urban design study," *Journal of ICT Research and Applications*, vol. 3, no. 1, pp. 1–18, 2009.

[110] "Unity solutions," https://unity.com/solutions, accessed: 2019-05-17.

[111] P. Coupé, P. Hellier, C. Kervrann, and C. Barillot, "Nonlocal means-based speckle filtering for ultrasound images," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2221–2229, 2009.

[112] F. Schmitt, M. Mignotte, C. Collet, and P. Thourel, "Estimation of noise parameters on sonar images," in *Statistical and Stochastic Methods for Image Processing, Denver, USA, October 8*, E. R. Dougherty, F. J. Preteux, and J. L. Davidson, Eds., vol. 2823. International Society for Optics and Photonics, 1996, pp. 2–13.

[113] A. Abu and R. Diamant, "Robust image denoising for sonar imagery," in *MTS/IEEE OCEANS Kobe Techno-Oceans (OTO)*, 2018, pp. 1–5.

[114] D. Kuan, A. Sawchuk, T. Strand, and P. Chavel, "Adaptive restoration of images with speckle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 373–383, 1987.

[115] F. Maussang, J. Chanussot, A. Hétet, and M. Amate, "Mean–standard deviation representation of sonar images for echo detection: Application to SAS images," *IEEE Journal of Oceanic Engineering*, vol. 32, no. 4, pp. 956–970, 2007.

[116] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[117] K. Perlin, "Improving noise," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 681–682.

[118] B. T. Henson, "Image registration for sonar applications," Ph.D. dissertation, University of York, 2017.

[119] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-Net: Learning of structure and motion from video," *arXiv preprint arXiv:1704.07804*, 2017.

[120] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, June 18-22*, 2018, pp. 1983–1992.

[121] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with CNNs for frame-to-frame ego-motion estimation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 18–25, 2015.

[122] H. Li and Y. Fan, "Non-rigid image registration using self-supervised fully convolutional networks without training data," in *15th IEEE International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 1075–1078.

[123] B. D. de Vos, F. F. Berendsen, M. A. Viergever, M. Staring, and I. Išgum, "End-to-end unsupervised deformable image registration with a convolutional neural network," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2017, pp. 204–212.

[124] D. Quan, S. Wang, M. Ning, T. Xiong, and L. Jiao, "Using deep neural networks for synthetic aperture radar image registration," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 2799–2802.

[125] S. Wang, D. Quan, X. Liang, M. Ning, Y. Guo, and L. Jiao, "A deep learning framework for remote sensing image registration," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 148–164, 2018.

[126] H. M. Keshk and X.-C. Yin, "Change detection in SAR images based on deep learning," *International Journal of Aeronautical and Space Sciences*, pp. 1–11, 2019.

[127] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, "Using synthetic data to train neural networks is model-based reasoning," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3514–3521.

[128] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[129] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[130] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 969–977.

[131] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," *Computer Vision and Image Understanding*, vol. 137, pp. 24–37, 2015.

[132] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," *arXiv preprint arXiv:1708.06020*, 2017.

[133] E. Richardson, M. Sela, and R. Kimmel, "3D face reconstruction by learning from synthetic data," in *IEEE Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA*, 2016, pp. 460–469.

[134] A. Kortylewski, A. Schneider, T. Gerig, B. Egger, A. Morel-Forster, and T. Vetter, "Training deep face recognition systems with synthetic data," *arXiv preprint arXiv:1802.05891*, 2018.

[135] X. Zhang, Y. Fu, A. Zang, L. Sigal, and G. Agam, "Learning classifiers from synthetic data using a multichannel autoencoder," *arXiv preprint arXiv:1503.03163*, 2015.

[136] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," *arXiv preprint arXiv:1702.07836*, 2017.

[137] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2315–2324.

[138] F. Zhan, H. Zhu, and S. Lu, "Scene text synthesis for efficient and effective deep network training," *arXiv preprint arXiv:1901.09193*, 2019.

[139] *Hovering Autonomous Underwater Vehicle Specifications*, Bluefin Robotics Corporation, 2015, available online: https://gdmissionsystems.com/-/media/General-Dynamics/Maritime-and-Strategic-Systems/Bluefin/PDF/Bluefin-HAUV-Datasheet.ashx (accessed on 08 July 2021).

[140] J. Vaganay, M. Elkins, S. Willcox, F. Hover, R. Damus, S. Desset, J. Morash, and V. Polidoro, "Ship hull inspection by hull-relative navigation and control," in *MTS/IEEE OCEANS, Washington, D.C., USA*, 2005, pp. 761–766.

[141] *ARIS Explorer 3000 Standard Version Specifications*, Sound Metrics Corp., accessed: 2021-01-28. [Online]. Available: http://www.soundmetrics.com/Products/ARIS-Sonars/ARIS-Explorer-3000/016621-A-ARIS-Explorer-3000-Specifications.

[142] T. Kessler, G. Dorian, and J. H. Mack, "Application of a rectified linear unit (ReLU) based artificial neural network to cetane number predictions," in *Internal Combustion Engine Division Fall Technical Conference*, vol. 58318. American Society of Mechanical Engineers, 2017, p. V001T02A006.

[143] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[144] S. Marschner and P. Shirley, *Fundamentals of Computer Graphics*. CRC Press, 2015.

[145] B. Henson and Y. Zakharov, "Estimating attitude and trajectory of forward looking imaging sonar using inter-frame registration," in *International Conference and Exhibition on Underwater Acoustics (UACE), Skiathos, Greece*, 2017.

[146] A. J. Hunter, S. Dugelay, and W. L. Fox, "Repeat-pass synthetic aperture sonar micronavigation using redundant phase center arrays," *IEEE Journal of Oceanic Engineering*, vol. 41, no. 4, pp. 820–830, 2016.

[147] L. Rixon Fuchs, C. Larsson, and A. Gällström, "Deep learning based technique for enhanced sonar imaging," in *5th International Conference and Exhibition on Underwater Acoustics (UACE), Hersonissos, Crete, Greece*, 2019, pp. 1021–1028.

[148] V. Myers, I. Quidu, B. Zerr, T. O. Sæbø, and R. E. Hansen, "Synthetic aperture sonar track registration with motion compensation for coherent change detection," *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 1045–1062, 2019.

[149] L. Shen, B. Henson, Y. Zakharov, and P. Mitchell, "Two-stage self-interference cancellation for full-duplex underwater acoustic systems," in *IEEE OCEANS Marseille*, 2019, pp. 1–6.

[150] L. Shen, B. Henson, Y. Zakharov, and P. Mitchell, "Digital self-interference cancellation for full-duplex underwater acoustic systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 192–196, 2019.

[151] A. Keary, M. Hill, P. White, and H. Robinson, "Simulation of the correlation velocity log using a computer based acoustic model," in *International symposium on unmanned untethered summersible technology*. University of New Hampshire-Marine systems, 1999, pp. 446–454.

[152] P. J. Boltryk, M. Hill, A. C. Keary, and P. R. White, "Surface fitting for improving the resolution of peak estimation on a sparsely sampled two-dimensional surface with high levels of variance, for an acoustic velocity log," *Measurement Science and Technology*, vol. 15, no. 3, p. 581, 2004.

[153] L. M. Wolff, E. Szczepanski, and S. Badri-Hoeher, "Acoustic underwater channel and network simulator," in *MTS/IEEE OCEANS, Yeosu, Republic of Korea*, 2012, pp. 1–6.

[154] M. Porter, "Bellhop3d user guide," https://usermanual.wiki/Document/Bellhop3D20User20Guide202016725.1524880335/html, (accessed on 01 June 2022).

[155] C. Liu, Y. V. Zakharov, and T. Chen, "Doubly selective underwater acoustic channel model for a moving transmitter/receiver," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 938–950, 2012.

[156] A. Song, J. Senne, M. Badiey, and K. B. Smith, "Underwater acoustic communication channel simulation using parabolic equation," in *Proceedings of the Sixth ACM International Workshop on Underwater Networks*, 2011, pp. 1–5.

[157] B. Henson, J. Li, Y. V. Zakharov, and C. Liu, "Waymark baseband underwater acoustic propagation model," in *IEEE Underwater Communications and Networking (UComms)*, 2014, pp. 1–5.

[158] "Matlab - regression learner app," Available online: https://mathworks.com/help/stats/regression-learner-app.html (accessed on 08 July 2022).