2022

Ship Motion Prediction with Deep Learning using IMU Data and Images

THIS IS A PLACE HOLDER TITLE PAGE

LANCE DE WAELE

(empty page)

(second title page)

(confidentiality notice)

Preface

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce ultricies, orci et scelerisque volutpat, nibh metus vestibulum ipsum, quis convallis ex orci ut massa. Curabitur felis dolor, tempor eu interdum nec, mattis quis felis. Vestibulum in nibh sit amet quam porta tristique. Fusce eu tortor tempus, tincidunt tortor hendrerit, sollicitudin elit. Cras a tempor urna. Vivamus vel malesuada purus. Sed feugiat egestas dolor, at feugiat lorem. Aliquam erat volutpat. Ut vel suscipit mi, quis vehicula lacus. Duis vitae libero semper, dignissim risus quis, vulputate augue. Praesent libero mauris, pretium id pharetra eget, malesuada et augue. Donec sed tincidunt augue. Nunc condimentum lectus non augue consequat, eget malesuada felis volutpat. Vestibulum ornare ultricies orci. Nulla sit amet dictum justo, non commodo arcu.

Abstract

(English)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce ultricies, orci et scelerisque volutpat, nibh metus vestibulum ipsum, quis convallis ex orci ut massa. Curabitur felis dolor, tempor eu interdum nec, mattis quis felis. Vestibulum in nibh sit amet quam porta tristique. Fusce eu tortor tempus, tincidunt tortor hendrerit, sollicitudin elit. Cras a tempor urna. Vivamus vel malesuada purus. Sed feugiat egestas dolor, at feugiat lorem. Aliquam erat volutpat. Ut vel suscipit mi, quis vehicula lacus. Duis vitae libero semper, dignissim risus quis, vulputate augue. Praesent libero mauris, pretium id pharetra eget, malesuada et augue. Donec sed tincidunt augue. Nunc condimentum lectus non augue consequat, eget malesuada felis volutpat. Vestibulum ornare ultricies orci. Nulla sit amet dictum justo, non commodo arcu.

Extended abstract

(Nederlands)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce ultricies, orci et scelerisque volutpat, nibh metus vestibulum ipsum, quis convallis ex orci ut massa. Curabitur felis dolor, tempor eu interdum nec, mattis quis felis. Vestibulum in nibh sit amet quam porta tristique. Fusce eu tortor tempus, tincidunt tortor hendrerit, sollicitudin elit. Cras a tempor urna. Vivamus vel malesuada purus. Sed feugiat egestas dolor, at feugiat lorem. Aliquam erat volutpat. Ut vel suscipit mi, quis vehicula lacus. Duis vitae libero semper, dignissim risus quis, vulputate augue. Praesent libero mauris, pretium id pharetra eget, malesuada et augue. Donec sed tincidunt augue. Nunc condimentum lectus non augue consequat, eget malesuada felis volutpat. Vestibulum ornare ultricies orci. Nulla sit amet dictum justo, non commodo arcu.

Table of contents

List of Figures

List of Tables

List of Abbreviations

AI: Artificial Intelligence
ASV: Autonomous Surface Vessel
IMU: Inertial Measurement Unit
PoV: Point of View
GPU: Graphics Processing Unit
LSTM: Long Short-Term Memory
CNN: Convolutional Neural Network
NN: Neural Network
ReLU: Rectified Linear Unit

Introduction

In the last few years, the world has seen an exponential increase in technological advancements. This evolution brought with it a new influence of autonomous systems controlled by artificial intelligence (AI). Each of these systems designed with its own goals and characteristics, optimized for its task. More and more of these systems are being deployed as a direct or indirect replacement for tasks humans could do, but also, for tasks humans can't physically do. And because these autonomous systems are optimized for specific jobs, they can be more accurate and faster at it than humans.

Because autonomous systems can replace the position of a human, they are especially useful in military operations. They can take over the role of a human in dangerous environments such as an active warzone and therefore eliminate the risk of someone's life. On the other hand, they can also be used as a complimentary asset, providing support and aid in logistics. More and more of these autonomous assets such as drones, surface vessels, tanks and reconnaissance vehicles are being deployed around the world for various objectives. But with this increasing amount of autonomous assets, there is need for communication between them, to allow them to work together and be aware of the state of each other when they need to interact (de Cubber, 2019).

"*Interoperability is the key that acts as the glue among the different units within the team, enabling efficient multi-robot cooperation."* (*MarSur – Robotics & Autonomous Systems*, n.d.)

*Figure 1: Prototype of Autonomous Surface Vessel (ASV) with on-board computer and sensors (*MarSur – Robotics & Autonomous Systems*, n.d.)*

The Robotics & Autonomous Systems lab of the Belgian Royal Military Academy is currently working on two autonomous vehicles in two projects named MarSur (*MarSur – Robotics & Autonomous Systems*, n.d.)and MarLand (*MarLand – Robotics & Autonomous Systems*, n.d.). Project MarSur is developing an autonomous surface vessel (ASV) (figure 1) that will interact with a drone that is being developed by project MarLand. The drone needs to be able to take-off and land on the ASV. This proposes a challenge since the ASV is continuously moving due to sea waves and can therefore not provide a stable landing surface. For a smooth landing to be possible, the ASV must be capable to determine its state in a three-

dimensional space and predict its movement in the ocean. These predictions must provide an accurate estimation over a future time series to determine the optimal time for the drone to land. This optimal time is a period where the ASV is as stable as possible so that the impact on the drone will be minimized. To facilitate these predictions, the ASV is equipped with an on-board computer and multiple sensors.

ASV sensors

The ASV is equipped with a ZED-mini stereo IMU camera (figure 2) (StereoLabs, Paris, France). This is a multipurpose sensor that can provide both video from its cameras and numeric data from its Inertial Measurement Unit (IMU), to accurately describe the state of the sensor and its surroundings. The ZED-mini has two built-in motion sensors: an accelerometer and a gyroscope. These provide a real-time data stream at 800Hz of the movement of the sensor along the rotational and translational axes. These types of movements will be described more in depth in the next part of the introduction.

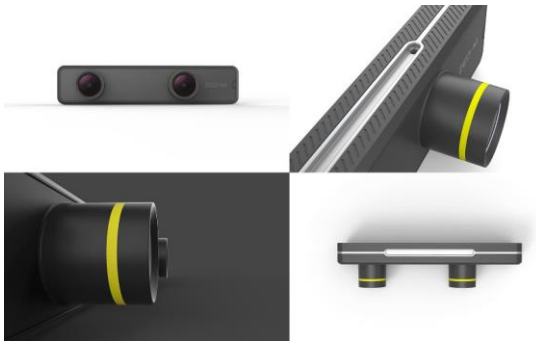

Figure 2: ZED-mini IMU stereo camera (StereoLabs, Paris, France)

The two forward facing lenses on the ZED-mini provide stereo video. This can be used to map the objects in front of the ZED-mini in a three-dimensional space. A stereo image is a combination of two separate images that are captured from two slightly offset point-of-views (PoV) such as the lenses on the ZED-mini. These two PoV's imitate the left and right human eye and create a perception of depth when the two images are fused together to create one stereo image. This process is called stereoscopy. In computer vision, these two images can be compared to each other to extract three-dimensional information from two dimensional images.

The data from the motion sensors and the camera's will be used as input to predict the future movement of the vessel. The deep learning models will digest a sequence of data and images of the past to predict a sequence of data in the future. To do this, the model will try to find trends in the data and continue these trends with regards to the information gathered from the images with incoming waves. Two important parameters will be how much historic data the model takes as input and how much data it can accurately predict in the future.

Ship motion in six degrees of freedom

The motion of a ship or any rigid object in a three-dimensional space can be described in six degrees of freedom. These six degrees can further be divided into two categories: translational and rotational movement. Where translational movement is movement along one of the three axes in a three-

dimensional space, rotational movement is the rotation of an object around these same three axes. These three reference axes run through the center of mass of the ship and are oriented as follows:

- Vertical Z-axis runs vertically through the vessel
- Transverse Y-axis runs horizontally across the vessel
- Longitudinal X-axis runs horizontally through the length of the ship

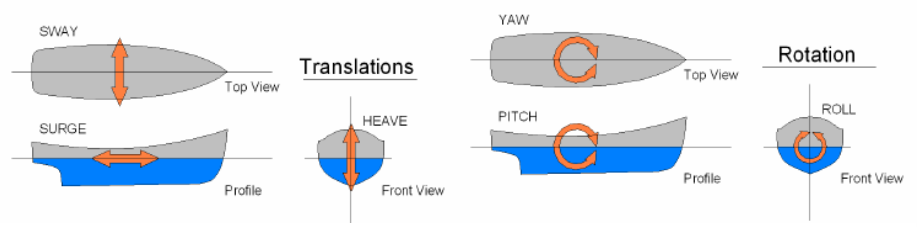

Figure 3: Six degrees of freedom in ship motion (de Masi et al., 2011)

Each type of motion, translation or rotation, among each of the three axes has a different impact on the movement of the vessel (figure 3). The translational movements are expressed in linear units such as meters and are named as followed:

- **Sway**: side to side movement along the transverse Y-axis
- **Surge**: forward and backwards movement along the longitudinal X-axis
- **Heave**: upward and downward movement along the vertical Z-axis

The rotational movements are expressed in angular units and are named as followed:

- **Yaw**: rotational movement around the vertical Z-axis
- **Pitch**: rotational movement around the transverse Y-axis
- **Roll**: rotational movement around the longitudinal X-axis

To predict the motion of a ship, one must differentiate between these different motions. Together they form the complete three-dimensional orientation of a ship. But not all of them need to be predicted. Surge and yaw are controlled by the ASV's autonomous systems and are respectively controlled by the amount of thrust and the rudder position – steering the ship. Surge and yaw will also not change very drastically during the landing or take-off of a drone since this behavior would directly impede our main goal of providing a smooth landing. On the other hand, the sway of a ship, also referred to as drift, is primarily caused by sideways winds or currents in the water and will have minor impact on the stability of the ship whenever the drone needs to take-off or land. If the drone aims for a GPS-tracker present in the ASV, it will follow the vessels movement no matter the sway.

This leaves three main factors remaining which have the most impact on the stability of the vessel: roll, pitch and heave. These three movements have one thing in common, they are all directly caused by the waves in the ocean and are very hard to control. Different methods exist to dampen these movements and keep the vessel as stable as possible such as bilge keels and antiroll tanks. However, most of them are either infeasible or ineffective or don't provide the required stabilization on smaller vessels (Perez & Blanke, 2017). In this case, predicting these movements instead of trying to dampen them, can be an alternative solution. Although it should be noted that using them together, will most likely yield the best performance. Pitch, roll and heave can be divided in two categories based on the effect they have on the landing and take-off of the drone. Pitch and roll are responsible for the stability of the landing surface and heave is responsible for the impact on the drone when landing.

To provide a stable landing zone for the drone, the pitch and roll of the vessel should remain constant and as close to zero as possible. Depending on the characteristics of the drone, the model should be able

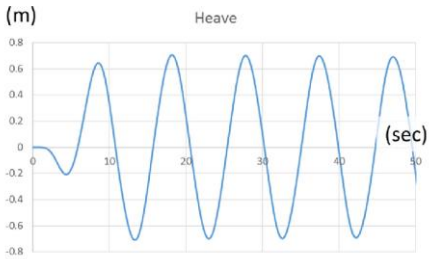

Figure 4: Heave (m) in function of time (s) (Ham et al., 2017)

to analyze its prediction sequence and find a window where the desired circumstances to land/take-off are met. To determine this window of landing/take-off opportunity, different parameters need to be defined such as the maximum difference in consequent prediction values, the length of the window and the interval in which all predicted values should lie. For example, the roll and pitch values should all remain in a [-3°, 3°] interval, the stable window duration must be at least five seconds and there should be no difference larger than two degrees between consecutive predicted values.

To minimize the impact on the drone when landing, the heave needs to be constant or decreasing. This means that the vessel is either not moving up or down, or it is slowly moving downwards following the motion of the descending drone. In regular waves, the heave of a vessel follows a wave-like function, alternating between upwards and downwards motion (figure 4). In this case, the window of take-off/landing opportunity can be defined as the points where the vessel transitions from upwards to downwards motion, or vice versa, and thus has an acceleration of zero.

Wave dynamics

Withing the boundaries of this thesis. Effect on deep learning models?

Technologies

To develop the deep learning neural networks, Python[1] will be used. Python is a high-level, general-purpose programming language that is in - most cases - most suited for machine learning applications. To facilitate the development of the deep learning models, different packages were used. The most important of which are briefly discussed below.

Met opmerkingen [LDW1]: Reference style of these kind of sources?

Pandas, Seaborn and NumPy were used to process and manipulate data. NumPy provides functionality to perform mathematical functions on large datasets with multiple dimensions. Pandas provides the functionality to visualize and order the data in its Data frames. Seaborn was mainly used to perform data analysis on the numeric data such as pitch and roll, to assess and visualize statistical information between features. To plot training and test results, Matplotlib was used.

[1] https://www.python.org/

For the implementation of the deep learning models, PyTorch was used. PyTorch is an open-source framework for machine learning applications that allows fast development and ease of use. Pytorch was used together with the cudatoolkit extension. This enabled the models to be trained and tested on a dedicated graphics processing unit (GPU) to increase computing performance.

Blender was used to generate a dataset from an ocean wave simulation. Blender is an open-source three-dimensional creation suite that can be used for many purposes. In this case, it was used to simulate incoming waves on a vessel. The data itself will be discussed more in detail bellow.

All code was written Jupyter Notebook for its ease of use and simple debugging. Whenever parts of the code in the notebooks were tested thoroughly, they are extracted to standalone Python files and accessed via import statements.

Simulation data

During most of the development, a simulation dataset was used from an ocean wave simulation made in Blender (*Nazotron1923/Ship-Ocean_simulation_BLENDER: Python Scripts for Generation Sea States and Ship Motion Using 3D Blender Simulator*, n.d.). This dataset was made by Nazar-Mykola Kaminskyi and is publicly available through GitHub. It was not made by or in cooperation with the research group of this thesis. Simulation data was needed for training and testing purposes during the initial phases of development when real data from the ASV was not yet collected and available.

For this simulation, a standard model of a vessel was used which is floating on the simulated sea surface and moves along with the waves. This presents the issue that all models trained with this data will be biased to this vessel's characteristics. A large, heavy vessel will behave very different opposed to a small, lightweight vessel. Therefore, all models should be retrained and re-evaluated with real data from the vessel it will be used on, as this vessel will most likely have different characteristics. However, this issue can also form a way to compare how well one model can predict the motion for different vessels. If there are only minor increases in performance when the model is retrained with data from the specific vessel, it might be more useful to use one general purpose model for all vessels with similar characteristics.

Using a simulation also presents a second issue: perfect conditions. The data taken from the simulation is from a perfect scenario, meaning that there are no obstructions or other objects on the images. The images are also perfectly stabilized on the moving vessel. This can cause models that are trained on the simulation data to perform very well yet fail to meet desired expectations in a real-world scenario. To minimize this effect, data can be augmented to include more variation, or adjustments can be made to the simulation. Adjustments such as including passing vessels in the images can make it more realistic.

Neural network architectures

Neural networks (figure 5), also known as artificial neural networks (ANN) are the building blocks of deep learning problems, which are a subset of machine learning problems and artificial intelligence. Neural networks are structures inspired by the human brain, more specifically the neurons within and how they pass signals from one to another.

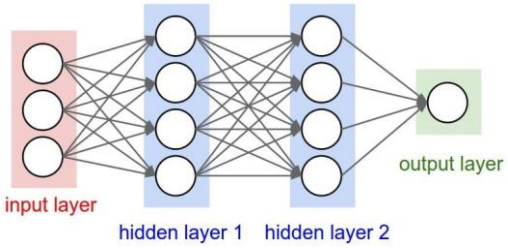

Figure 5: structure of a basic neural network

A neural network consists of different layers built with multiple neurons (Figure 5). These neurons are connected to neurons and pass data forward over these connections. The output of one neuron is the input of all the neurons it connects to. This way data is fed forward through the network and updated in every neuron. There are three main parts in a neural network, the input layer, output layer and hidden layers. The input layer has the same number of neurons as the features in the input data. The output layer has the same number of neurons as the predicted output features. In our case, the output features will be pitch, roll and heave.

Different architectures exist for different applications, each with their strengths and weaknesses. In this thesis, sequential data will be used which contains numeric data as well as images. Because of this, different architectures will be used that perform best with this kind data. Since no single model architecture exists that can handle all data well, different architectures will be combined to form hybrid models. Each of the used architectures will be discussed more in detail in the following sections.

Auto-encoders

TODO

Long Short-term Memory networks

TODO

Convolutional Neural Networks

TODO

Other

TODO

1 Related work

A big source of information for this this thesis was the work of Nazar-Mykola Kaminskyi (Kaminskyi, 2019). Kaminskyi also tried to create different neural networks that can predict the motion of a vessel

based on pitch, roll and incoming wave images. He created the dataset that was used for a big portion of this thesis and provided research and results from his research project. His results were used to set up criteria to which our models had to comply. This thesis should not be seen as a replacement for Kaminskyi's work though, but more as continuation and expansion of his work. Kaminskyi never tested his models on real life data and only used pitch and roll as input together with the images. With the ZED-mini, our models will have access to more data and should therefore potentially perform better.

TODO: discuss his results and how this will affect the methods in this thesis.

2 Objectives

The main goal of this research is to develop a neural network capable of predicting the motion of a surface vessel. To do this, the model ingests a sequence of values describing the state of the vessel and images of incoming waves taken from a stabilized camera pointing to the front of the vessel. As an output, the model should provide a sequence of predicted pitch, roll and heave values for every predicted second. The sequence duration for these predictions should be at minimum thirty seconds to provide the drone with ample time to complete a take-off or landing procedure. Bigger drones will need more time for this procedure so the sequence duration should be maximized within the model's capabilities.

Since the model will be deployed in real-time scenario's and will be making predictions in real-time based on the continuous data stream of the ZED-mini, it should be lightweight and not require large amounts of computational power. This means that the inference time of the different proposed models should also be considered when comparing different models' performances. Both prediction errors and inference time should thus be minimized.

To test the viability of each model, it should be subjected to a set of minimum criteria in the simulation environment. A model is fit to be used with real data only when it meets these criteria. To make this selection, the following attributes of the model must comply with their respective requirements. The root mean square error (RMSE) for the predicted angles for pitch and roll at the 10^{th} second should be no more than three degrees and no more than five degrees at the 30^{th} predicted second. These values are derived from the performance of the best models from Kaminskyi's work. The RMSE of the heave of the vessel should never exceed twenty centimeters. Finally, the inference time of the model should be no more than fifty milliseconds.

Met opmerkingen [LDW2]: Cite

Whenever a prediction sequence is calculated, it should be analyzed by an algorithm. This algorithm will search the sequence for a window of landing/take-off opportunity. This window is a ten second time frame. However, the duration of this window should be easily changeable, to compensate for different sized drones. During this window, heave, roll and pitch should remain close to constant. To define this constant behavior, the next set of criteria will be used. Pitch and roll angles should remain in a [-3°, 3°] interval and consecutive values should have no difference bigger than two degrees. Consecutive heave values should have no difference bigger than one centimeter. For heave, there is no interval in which all predicted values should remain since this is highly dependent on the height of the waves.

3 Data

The first step of any machine learning operation is collecting data. For this thesis, data was collected from two sources as discussed in the introduction. A dataset from simulation and real-world data was used. This poses some challenges as the simulation data is captured in an ideal scenario and contains less motion parameters as the real-world counterpart. This can cause the models to perform differently and have slightly different architectures due to the different inputs and outputs based on the available data. In the following two sections, both datasets will be discussed in more detail. For the remainder of this thesis, the different parameters such as pitch, roll, yaw etc. will be referred to as the *features* of the dataset.

3.1 Simulated data

The simulated dataset contains images of incoming waves (figure 6) and the pitch and roll values of a vessel floating on these incoming waves. The data was generated in 540 different episodes, each episode containing 400 frames of data. The frames were captured at two frames per second to minimize data overlap in consecutive images. Each frame contains one image together with the state of the vessel at the time of the image in the form of a pitch- and roll-value tuple. In its totality, this dataset contains 216.000 frames, which translates to thirty hours of simulated data at two frames per seconds. All episodes were structured in the same way. Images were named 1 to 400 and pitch and roll data points were included in a *JSON* file. The pitch and roll couples were numbered 1 to 400 as well to easily identify which image corresponds to which tuple.

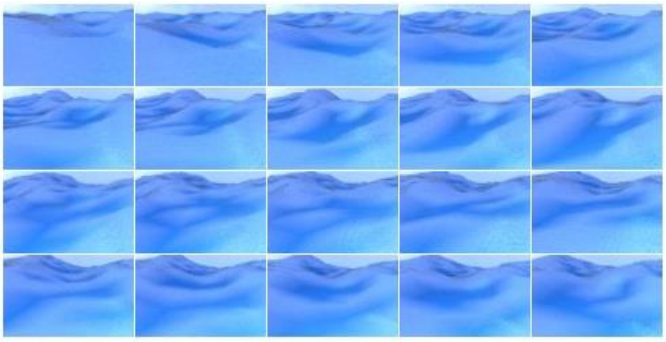

Figure 6: Generated images of incoming waves

With this simulated data we only have access to pitch and roll as input data. This means that during testing in the simulation environment, heave cannot be predicted. However, with the neural networks that will be proposed later, this should not be a big issue for two main reasons. First off, heave follows a somewhat predictable pattern in regular conditions as depicted by figure 4. Because of this, it was assumed not necessary to regenerate the full dataset for just one extra output. It was also assumed that if the model performed well on pitch and roll, it should consequently perform well on heave. Secondly, adding extra input and/or output features to a neural network architecture, is not an expensive operation. All models will have to be retrained anyway when switching from the simulation environment

to a real-world environment to adapt to the new vessels' characteristics and the additional data available from the ZED-mini.

3.1.1 Simulation parameters

To simulate this data, different parameters were used to finetune the simulation environment. The effects of these parameters on the simulation were not tested as the used dataset was already generated with fixed parameters. However, assumptions can be made on how they would affect the simulation. The most important parameters will be discussed briefly.

As mentioned above, the images were taken at two frames per second by a virtual camera. The position of the camera on the simulated vessel was set with the following parameters:

- Height: *5 meters*
- Rotation around x-axis: *76 degrees (slightly tilted downwards)*

The images were captured in a low resolution to decrease the memory needed to load the dataset. The following resolution parameters were used:

- Height: *54 pixels*
- Width: *96 pixels*

3.2 Real data

At this moment, data has been captured from the ASV but is not yet available.

The real data will be fed into the model at real-time. This means that the data stream will have to be cleaned and filtered to the desired input format for the model. Numeric motion parameters such as pitch and roll, are output by the ZED-mini at 800Hz. This data stream will have to be reduced to minimize data overlap. Since the simulation data was generated at two frames per second, 2Hz should be a good starting frequency to test the models. Later on, this frequency can be increased or further decreased depending on the model's performance and inference time.

Similarly, the video framerate from the stereo camera should be reduced to minimize data overlap between images and reduce computational requirements. The video footage will also have to be compressed to reduce the memory needed to process the images.

(From this point in the thesis onward, only the simulation data is used)

3.3 Data pre-processing

Before the images and numerical data can be loaded into the model, they needs to be processed. This pre-processing of data is necessary to enhance to performance of the models and ensure that the model operates correctly. Due to a simulation dataset being used, little pre-processing is needed since the data is collected in a predefined format. In the case of the ZED-mini, the data-stream needs to be reduced and cleaned as discussed in section 3.2.

3.3.1 Data cleansing

The first step of data pre-processing is data cleansing. In this step, data points are checked for inaccuracies and corrupted values. Odd data points are corrected or removed, and unnecessary features are dropped from the dataset.

In the case of the simulated dataset, this task is rather short as the data was generated following a specific format as discussed in 3.1. To cleanse the simulated data, all episodes were evaluated to contain 400 images and 400 pitch and roll couples. Next, all pitch and roll values were audited to not contain any odd values such as un-numerical values or odd outliers. This was done with *Pandas.isnull().values.any()* and *Pandas.DataFrame.plot.scatter()*. The scatter plot (Figure 7) shows the 2D lay out of pitch and roll. The data is mainly concentrated in the middle with few outliers. Further analysis of the distributions will be conducted in 3.4.

Met opmerkingen [LD3]: Is this expected or can it be left out.

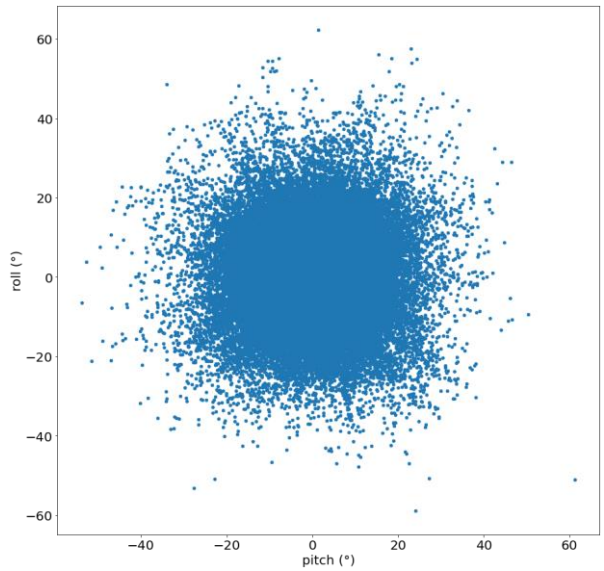

Figure 7: Scatterplot of pitch and roll

When real data is used, comprehensive cleaning is needed to filter out noise from the sensors and excessive outliers from inaccurate readings. *(TODO: add real data cleansing process)*

3.3.2 Data formatting

All data features fed into a neural network must be formatted in the same way. This means that all features containing numerical data should be converted into the right datatype and images should be resized to the same resolution. Textual data should also be pre-processed but this falls outside the scope of this thesis as no textual data will be used. This formatting is very important as incorrectly formatted data cannot be used to train and test neural networks.

For the simulation data, little formatting had to be done. All images were generated with the same resolution, so image resizing was not necessary. The pitch and roll values were imported from JSON-files as textual string data. They were subsequently casted to 64-bit floating point values for high accuracy.

3.3.3 Data reduction

Real data only: resizing/cropping/compressing, filtering 800Hz data and 60fps video to 2Hz

3.3.4 Data normalization

When one feature contains values in a range of [0, 100] and another feature contains values in a range of [0, 5], the first feature will have a much larger impact on the result if they are used together. For this reason, data needs to be normalized.

Pitch and roll are normalized with min-max feature scaling. This method rescales values relative to their absolute maximum and minimum within a [-1, 1] range. Since pitch and roll define the tilting of a ship, the absolute theoretical minimum and maximum were chosen to be -90° and 90° (Equation 3.1). If a ship pitches or rolls beyond these values, it will capsize. It is also not possible to the define the minimum and maximum as the boundaries of the measured values in the dataset, because these numbers are not known when working in real-time.

$$x_{normalized} = \frac{x_{original} - (-90)}{90 - (-90)} \cdot 2 - 1 = \frac{2 \cdot (x_{original} + 90)}{180} - 1$$

Equation 3.1: Min-Max normalization of angular motion parameters pitch and roll

Each images needs to be normalized as well. Each image is three-dimensional matrix containing three values - the red, green and blue channel - for each pixel in the image across the length and width. This means that for every pixel, three values need to be normalized. The three channels indicate the amount of red, green and blue in each pixel. These values are 8-bit integers, meaning that they range from 0 to 255. To normalize them, a similar min-max feature scaling function was used as for the pitch and roll values (Equation 3.2).

$$p_{normalized} = \frac{p_{original} - 0}{255 - 0} \cdot 2 - 1 = \frac{2 \cdot p_{original}}{255} - 1$$

Equation 3.2: Min-Max normalization of RGB-values

TODO: normalization for real data: images (cropping), thrust, wind velocity, heave? (m)…

3.4 Data analysis

The simulation data was briefly analyzed to have a better understanding of the results. Very in-depth analysis of the input data in not necessary for deep learning applications. Deep learning is a form of unsupervised learning, where the model itself determines the importance of each feature. In comparison, supervised learning problems require thorough data analysis to find out which feature(s) have most impact on the desired output feature(s) and should be included in the models' input. The goal of the data analysis in this thesis is to firstly better comprehend how they behave and secondly the interactions between pitch and roll.

3.4.1 Statistical properties

First off, basic statistical information was extracted from all pitch and roll data points from all episodes. With the built-in function of Pandas pd.DataFrame().describe(), this is easily achieved. The results are shown in

Table 1.

	PITCH	ROLL
count	216.000	216.000

mean	0,0678°	0,303°
standard deviation	6,611°	7,022°
minimum	-53,721°	-58,846°
25%	-2,761°	-2,572°
50%	0,0262°	0,230°
75%	2,876°	3,187°
maximum	61,328°	62,217°

Table 1: Statistical information for pitch and roll in simulated dataset

Met opmerkingen [LD4]: Fix formatting

The count refers to the amount of datapoints analyzed which is equal to the 540 episodes, containing 400 frames each. From these values, it can be concluded that pitch and roll both behave similarly on a numerical basis across all fields. However, this does not necessarily mean that they physically behave similar. The minimum and maximum values give a good idea of the interval in which the predicted values should remain. They also provide a means to normalize the RMSE. An error of three degrees is a 5% error and thus really good when all data occurs in a [-60°, 60°] interval. If all data lies in a [-10°, 10°] interval, three degrees is a 30% error which is very high. But the minimum and maximum don't tell the full story.

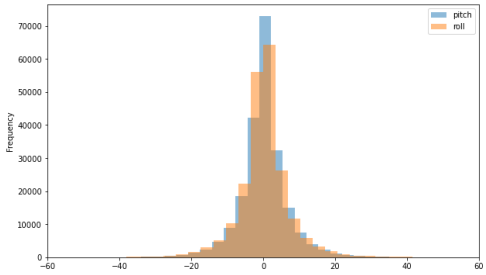

Figure 8: Simulated Pitch and Roll distributions

The distributions of pitch and roll are shown in Figure 8. Both of them show very similar characteristics as a normal distribution. Intuitively, it can be assumed that the majority of the datapoints are located in a [-20°, 20°] interval. Compared to the properties of normal distribution, this assumption is justified when the 65%-95%-99.7% rule is applied. This rule determines that given a mean **μ** and a standard deviation **σ** of a normally distributed dataset, respectively 65%, 95% and 99.7% of all datapoints lie within a **μ ± σ, μ ± 2σ** and **μ ± 3σ** interval. If this rule is applied with the mean and standard deviation from Table 1, the 99.7% interval can be calculated and are the following (values rounded down for readability):

- Pitch: *[-19,74°; 19,86°]*
- Roll: *[-20,70°; 21,30°]*

These intervals confirm that almost all data points – 99.7% – lie within the above intervals. Because of this, the RMSE should be taken relative to these interval boundaries rather than the maxima and minima of the full dataset. The 99.7% intervals represent almost all data whereas the minima and maxima could be two extreme outliers.

3.4.2 Correlation and interactions

Secondly, the correlation of pitch and roll was reviewed to assess the influence of roll on the prediction of pitch and vice versa. This was done to decide if having both features as input, has a positive effect on the predictions of individual the individual features.

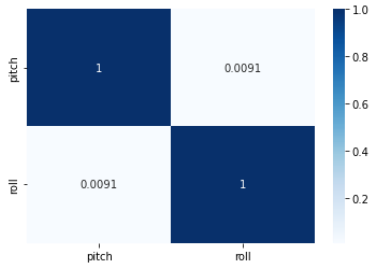

Figure 9: Correlation matrix for pitch and roll

A commonly used method for analyzing correlation is a correlation matrix. This is a symmetric matrix where the number of rows and columns equals the number of features. Each cell in the matrix contains a value equal to the correlation of the features of the corresponding row and column. These values range from -1 to 1 where -1 indicates a strong negative correlation, 1 a strong positive correlation and 0 indicates that there is no correlation. From the matrix in Figure 9, pitch and roll have a correlation close to zero, which indicates little to no correlation.

With this result, it can be assumed that one feature has negligible contribution to the prediction of the other feature. Providing additional features besides the predicted feature can therefore be seen as overhead for the model. However, this assumption is only based on numerical values from a simulation containing only pitch and roll. According to this research (Nayfeh et al., 2012), pitch and roll are coupled nonlinearly when their frequencies are in a ratio of two to one. On top of this, more correlated features could be introduced with the additional input features from the ZED-mini sensor.

With this information, it was concluded that standalone feature correlation was not enough evidence to rule out other possible physical interactions. Because of this, all models were designed to ingest all available features instead of dropping some due to low correlation with the predicted target features. In the case of the simulation data, this meant that models were trained and tested to predict both pitch and roll from an input sequence also containing both pitch and roll.

3.5 Data loading

Data loading is the process of creating input-output sequences, splitting the data and . For each model, the inputs and outputs will be sequences with varying lengths. Next, these generated sequences will be split into three subsets for training, testing and evaluating the model. Finally, the data will be wrapped in PyTorch DataSet and DataLoader objects which will convert the data into Tensors and create batches.

3.5.1 Sequence creation

The simulated data is generated in episodes. Each episode containing 400 frames that chronologically follow each other as can be seen in Figure 6. However, the episodes themselves do not follow each other chronologically. Different episodes are generated in different simulation sessions and should therefore not be seen as continuous. Because of this, the data must be sequenced correctly to not include data from different episodes when used for deep learning models that ingest and predict chronological sequences. In short, all frames in the input and output sequence, should always be from the same episode.

For this reason, a dedicated function was designed to create sequences from a given dataset. One sequence consists of two parts, an input sequence and an output sequence. The input sequence is the data used to predict the output sequence. The length of the input and output sequence can be passed to the function as a parameter to easily create new sequences. This is necessary as the length of the input and output sequences will have a large effect on the performance of the model and should be thoroughly examined. In addition, the function also accepts parameters to define the input and output features when different datasets are being used.

Met opmerkingen [LD5]: Provide code?

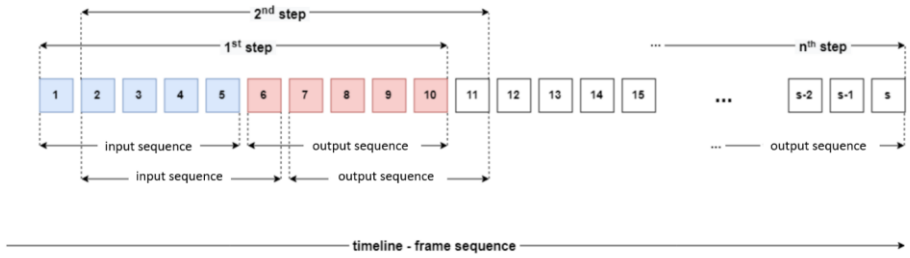

Figure 10: Sequence creation with moving input and output sequence

In Figure 10, the sequence creation process is illustrated with input and output sequences having a length of five. In the first step, the function takes the first five elements of a given episode with length **s**. These five elements form the input sequence, which will be used to predict the next five elements: the output sequence. These two sequences are then coupled together and added to a list with all sequences. In the second step, the starting point for the sequences is moved to the next element and the same procedure is carried out, extracting and coupling an input with an output. This process is repeated until the end of the episode is reached. At this point, the function moves to a new episode and repeats the cycle. This way, all input and output sequences can only contain consecutive data from the same episode.

3.5.2 Train-test-evaluation split

After the sequencing is done, the list of all coupled input-output (IO) sequences is split up in three subsets. Each subset will be used for a different purpose: one for training, one for testing and one for evaluating the models. This splitting is necessary to properly test and evaluate the model's performance.

When a machine learning model is trained on a dataset, it will try to optimize its performance as much as possible on this data. Therefore, in order to evaluate how well the model generalizes to new data, a second dataset is needed to test the model. This test dataset contains data that the model hasn't seen

before and is not optimized towards, unlike the training data. So, if the model is tested on this new dataset, the real performance on new data can be evaluated. After this evaluation with the test dataset, the models' parameters can be adjusted in the hopes of further optimizing the generalization to new data. Finally, a third dataset is used to measure the performance of the model, after being trained on training data and optimized with test data. After this, no changes are made to the model to ensure that the results from the evaluation dataset, represent the model's true core performance on new data.

To achieve this split, elements from the list of coupled IO-sequences are randomly selected and assigned to one of the three subsets. This randomization has positive effect on the model's performance because it mitigates possible trends in the initial dataset. For example, due to pure coincidence it is possible that the first ten IO sequences are from calm seas and the next 5 are from rough seas. If this dataset is split using standard iteration over the data - adding the first ten IO-sequences to training and the next five to testing - the model will perform very poorly as it never had the chance to learn the rougher seas trends. Thus, the data is split with random selection.

The proportions of each subset are relative to all available data points, i.e. IO-sequences. For most of the research, the following data split was used:

- Training dataset: *70%*
- Test dataset: *15%*
- Evaluation dataset: *15%*

3.5.3 DataLoader and DataSet

Finally, after the pre-processed data is sequenced and split, it is prepared for the model. All neural networks were designed in PyTorch. Because of this, it was most convenient to load the data with a PyTorch DataLoader. The three different datasets – training, testing and evaluation – are wrapped inside a DataLoader object. The DataLoader in turn returns this data as an iterable PyTorch DataSet object and splits it into batches.

The DataSet class acts a data structure which holds the IO-sequences. It is a custom class which inherits from the PyTorch DataSet class and overrides its main method: *__init__()*, *__len__()* and *__getitem__(int)*. The latter of these methods returns the IO-sequence at the given index as a dictionary. This dictionary holds two key-value pairs – one for the input and one for the output sequence - converted into a PyTorch Tensor. A Tensor object is an n-dimensional array that allows functions to be carried out over all elements at once, much like a NumPy array.

The DataLoader class provides methods to load training, test and evaluation data. Part of this loading process involves splitting the data into batches with a fixed batch size. The size of the batches is an important hyperparameter and determines how much data the model will consume before updating its internal parameters. Batch size and its effects will be discussed more in detail in section 5.1.

In conclusion, data is wrapped inside a DataLoader object, which in turn wraps the data in a DataSet object and divides it in batches. The DataLoader can then be called to load data and returns this as an iterable object, with each batch of **n** samples being one iteration.

4 Model designs

In this chapter, all models will be discussed that were created. An iterative process was used where each model design tries to improve on its predecessor. As discussed in the introduction, hybrid models will be used to obtain optimal performance with the given data and problem definition. To identify different models based on their architecture, the following notations are introduced ().

Notation	Explanation
P, R, PR	Pitch, roll or Pitch and roll were used as input/output
LSTM	A LSTM architecture was used
CNN	A CNN architecture was used
Multivariate	Multiple input features were used
Single-/Multi-step	Single value output/sequence of values output

4.1 Multivariate single-step LSTM models

TODO: add architecture illustration (software?)

4.1.1 Stacked single-output LSTM

The first two models that were created, were multivariate stacked LSTM architectures using both pitch and roll as input. Two models were created, one to predict and pitch one to predict roll. The models use a sequence of pitch and roll This means that they can only predict one frame of one variable: either pitch or roll. These models were designed to get familiar with the workflow of designing, training and testing neural networks with PyTorch.

Design parameters of this model were arbitrarily chosen. A two layered or stacked LSTM configuration was used where the output of the first LSTM is used as input by a second LSTM. Both of these LSTM's used 128 hidden neurons and dropout of 0.2. The last layer is a linear fully connected layer, combining the 128 hidden features to a single output feature.

(architecture image)

4.1.2 Stacked multi-output LSTM

For this model, a slight adjustment was made to the above model. The number of output features was increased to two. This way pitch and roll could be predicted simultaneously from a sequence of pitch and roll.

(architecture image)

4.2 Multivariate Multi-step LSTM models

4.2.1 Encoder-decoder LSTM

The next step in development was to expand the capabilities of previous model to be able to predict a sequence of pitch and roll. With this model, it was possible to examinate how well the model could predict pitch and roll with having the images on incoming waves. If this model would perform similarly to one which used images as an additional input, the conclusion could be made that, if necessary, the images could be omitted. In this case, this model would provide a more lightweight solution, requiring only numeric data.

Two LSTMs are stacked in this configuration. One acting as an encoder and one acting as a decoder. To predict a sequence, the encoder ingest the input features and passes its hidden and cell state forward to the decoder. The decoder has two fully connected layers at the end, the first one reducing the output to half of its input and the second reducing the halved output to two (pitch and roll).

The encoder and decoder LSTM both used a hidden layer size of 300 and are double stacked LSTMs with a dropout of 0.2.

(architecture image)

4.3 CNN models

4.3.1 Simple CNN

This model was created to test the performance of a model using only images as input to predict the pitch and roll. A sequence of images would be taken as input to predict a future sequence of pitch and roll.

5 Training

5.1 Parameters

TODO

Following parameters were used:

- Batch size: 64
- Epochs: 8
- Learning rate: 0.001

5.2 Optimizer

TODO

5.3 Loss function

TODO

5.4 Hyperband

TODO

5.5 Early stopping

TODO

6 Testing and Evaluation

To test the different models and tweak their parameters, the test dataset was used. After which the evaluation dataset was used to evaluate the core performance. However, hyperband optimizations and parameter tweaking was only done to push the best models to their optimum. The simple models were

mostly left in their initial state since newer iterations were used to improve them rather than small parameter tweaks.

To analyze the performance of the models, mainly the MSE, RMSE and graphs depicting original vs. predicted were used. As a baseline, each model was compared to a *zero-predictor*. This is a model that predicts zero every time and follows no trend. This was done to evaluate whether the models captured the trends of the data well.

6.1 Stacked single-output LSTM vs. Stacked multi-output LSTM

Firstly, both single-step models will be compared. These models should perform very well as they only need to predict one value. The three models were trained and tested with an input sequence of 50 PR-values over 8 epochs.

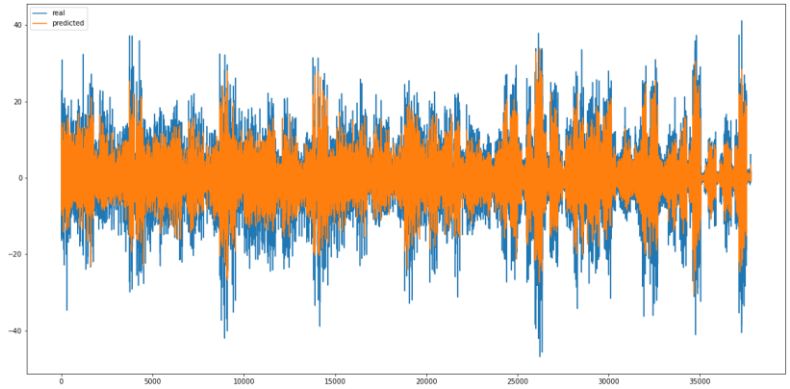

Figure 11: Predicted pitch vs. real pitch

Figure 11 shows the predicted pitch in orange on top of the real value in blue for all datapoints in the test dataset from the single-output model. This graph shows that the predicted pitch values seem to follow the trend very well but fall short of the real values by a couple degrees. It is important to note here that since the test dataset was constructed by random selection as discussed in 3.5.2, the values on this graph are not in are not chronological order and are from different episodes. However, it is still possible to draw conclusions on how well the model can follow the trend in a sequence to predict a future value since otherwise, the orange plot would not be mimicking the blue plot. This holds true for all future graphs of this type.

Figure 12 shows the same graph for the single output for roll. Similar conclusion can be made although it seems that roll is slightly easier to predict. Where pitch predictions were lacking a couple degrees, roll predictions overlap more with real values and should therefore have a lower RMSE.

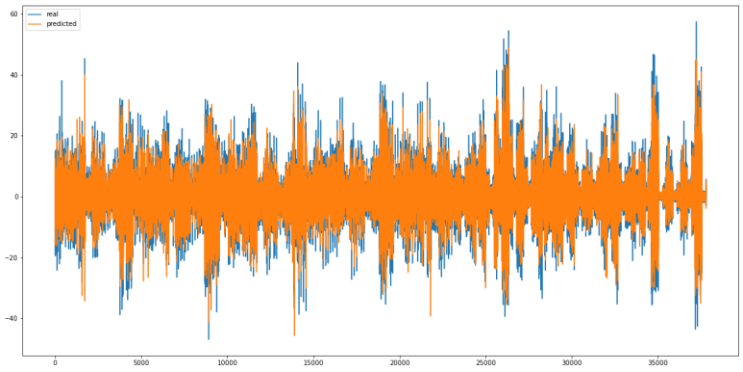

Figure 12: predicted roll (orange) vs. real roll (blue)

With these results in mind, the multi-output model's performance can be analyzed and compared. Figure 13 and Figure 14 show the prediction vs. real graphs for respectively roll and pitch. The dark areas sand-colored areas overlap well, orange or blue areas means that there are differences. When comparing the results, the two types of models have almost identical results. The predicted pitch values again fall short of the real values whereas the roll values have a closer overlap.

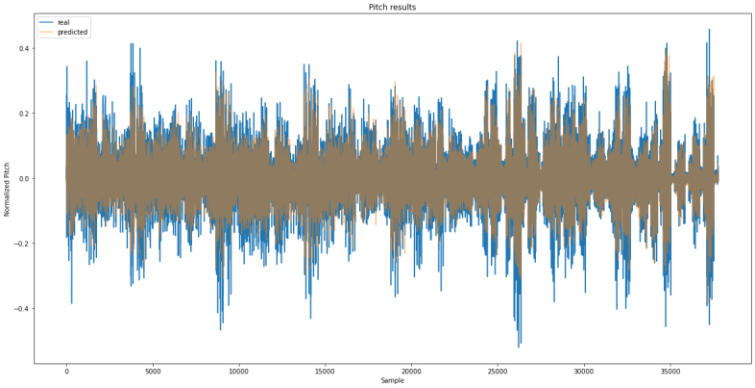

Figure 13: predicted pitch (orange) vs. real pitch (blue)

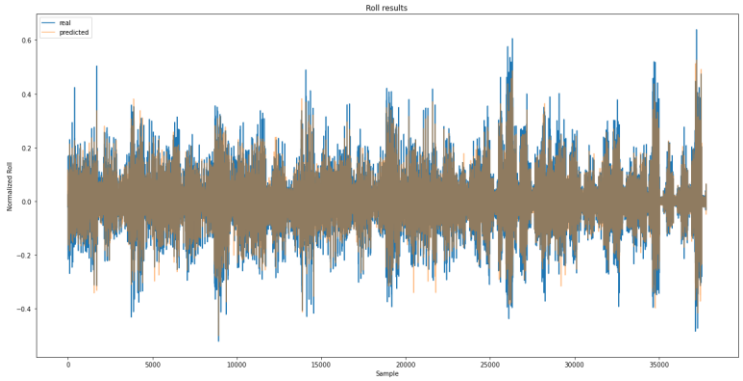

Figure 14: predicted roll (orange) vs. real roll (blue)

When comparing the errors of both model types, the observations from the graphs are confirmed. Table 2 shows the denormalized RMSE values for each model. The multi-output model performs slightly better than the single-output models one row one and two. On the bottom row the RMSE for a zero prediction was added. As expected, these values are significantly worse than the trained models. This means that the models have indeed captured some form of underlying trend in the data and are making logical predictions.

On the 4th and 5th row, the results were added of the multi-output model when trained for more epochs and used with lower input sequence length. This was done to assess the effect of these parameters and to find an optimal input length.

MODEL (in -> out - epochs)	PITCH (denormalized RMSE)	ROLL (denormalized RMSE)
50x2 PR -> 1x1 Pitch - 8	4.69°	/
50x2 PR -> 1x1 Roll - 8	/	3.76°
50x2 PR -> 1x2 PR - 8	4.40°	3.54°
50x2 PR -> 1x2 PR - 50	2.17°	2.03°
10x2 PR -> 1x2 PR - 50	2.34°	2.09°
Zero prediction	6.43°	7.30°

Table 2: denormalized RMSE for single-output LSTM models in different configurations

Increasing the number of epochs allows the model to run through the full training dataset more times. Therefore, it will update its internal parameters more and should be better performing. This show by the results in the above table. When increasing the number of epochs to 50, the RMSE values for pitch and roll both lower down to around 2°, which is a big improvement. The same configuration with 50 epochs was also tested with a lower input sequence. This was done to find out whether or not it is necessary to provide 50 frames. From the results, only a small increase in RMSE is shown.

Finally, the generalization of the multi-output model was analyzed by plotting the MSE values the model per epoch. This is shown in Figure 15. From the trend of the graph, it is clear that the MSE is still decreasing at 50 epochs. This means that we either can train the model for more epochs to further increase is performance, or that this model has a poor generalization and is maybe overfitting.

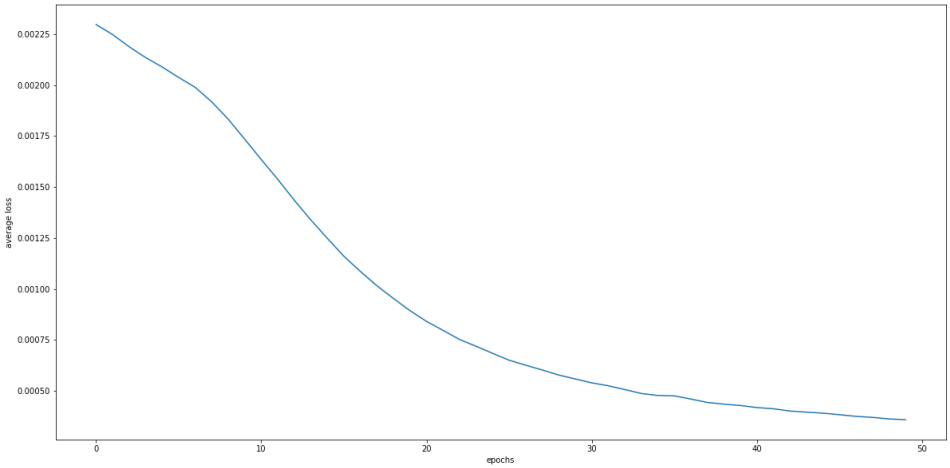

Figure 15: MSE of the multi-output single-step LSTM model in function of epochs

6.2 Encoder-decoder LSTM

No results yet

6.3 CNN

No results yet

7 Conclusion

TODO

7.1 Future improvements

TODO

8 References

de Cubber, G. (2019). *OPPORTUNITIES AND SECURITY THREATS POSED BY NEW TECHNOLOGIES*.

de Masi, G., Gaggiotti, F., Bruschi, R., & Venturi, M. (2011). Ship motion prediction by radial basis neural networks. *IEEE SSCI 2011 - Symposium Series on Computational Intelligence - HIMA 2011: 2011 IEEE Workshop on Hybrid Intelligent Models and Applications*, 28–32. https://doi.org/10.1109/HIMA.2011.5953967

Ham, S.-H., Roh, M.-I., & Zhao, L. (2017). *Integrated method of analysis, visualization, and hardware for ship motion simulation*. https://doi.org/10.1016/j.jcde.2017.12.005

Kaminskyi, N.-M. (2019). *Deep learning models for ship motion prediction from images*.

MarLand – Robotics & Autonomous Systems. (n.d.). Retrieved March 29, 2022, from https://mecatron.rma.ac.be/index.php/projects/marland/

MarSur – Robotics & Autonomous Systems. (n.d.). Retrieved March 29, 2022, from https://mecatron.rma.ac.be/index.php/projects/marsur/

Nayfeh, A. H., Mook, D. T., & Marshall, L. R. (2012). Nonlinear Coupling of Pitch and Roll Modes in Ship Motions. *Https://Doi.Org/10.2514/3.62949*, *7*(4), 145–152. https://doi.org/10.2514/3.62949

Nazotron1923/ship-ocean_simulation_BLENDER: Python scripts for generation sea states and ship motion using 3D Blender simulator. (n.d.). Retrieved March 29, 2022, from https://github.com/Nazotron1923/ship-ocean_simulation_BLENDER

Perez, T. ;, & Blanke, M. (2017). Ship Roll Damping Control. *Annual Reviews in Control*, *36*(1), 129–147. https://doi.org/10.1016/j.arcontrol.2012.03.010

9 Appendix