Source Detection Performance Comparison between Potential and Turbulent Flow:

An Artificial Neural Network Approach
Bachelor's Project Thesis

Jonathan E. Reid
Supervisors: Sietse M. van Netten, Primoz Pirih, Ben J. Wolf

Abstract: In this paper, we compare the performance of several neural networks on underwater source detection using an artificial lateral line on realistic simulated flow data. Previous work has shown how the fish lateral line helps fish determine a source's location in water. Based on this, a number of studies using potential flow data have shown neural networks to be able to perform source location and source angle estimation satisfactorily. In the present study, we aimed to recreate those results using more realistic turbulent flow data generated by using a three dimensional Stam-type fluid solver to simulate flow produced by a sphere moving through a two dimensional plane of fluid. The more realistic data did indeed prove to be more difficult for the neural networks. Even so, the source location estimation suffered a loss in accuracy of only a factor of two, while the angle estimation suffered a loss in accuracy of a factor of five.

1 Introduction

Fish possess a sensory organ known as the lateral line, which allow them, amongst other things, to localise moving and vibrating sources in their direct environment (Dijkgraaf, 1963). Research has been done to determine how exactly the flow of water around a fish allows a fish to engage in these complex behaviours (Curcic-Blake and van Netten, 2006).

The fish sense fluid velocity profiles, which is a concatenation of the fluid velocities measured at different points along the lateral line. This information is then further processed by a neural network in a way that allows them to determine a number of high-level details about the objects in their surroundings. Treating the fluid velocity profiles as input, and the information about the objects as output, we can see that in between there must be a mapping of input to output. Artificial neural networks (ANN) are based on the functioning of neurons, and so, using a suitable ANN implementation, we may be able to solve this problem sufficiently well.

Previous work using potential flow data has shown a number of neural networks to be proficient in localising a source on a two dimensional plane using an artificial lateral line, which is essentially an array of sensors, which register the fluid velocities at certain points (Boulogne, Wolf, Wiering, and van Netten, 2017). Potential flow was chosen as the model because of its computational ease and time-invariant predictable behaviour. A downside to using potential flow is that it doesn't exhibit all traits of flows encountered in the real world, and direct generalisation of those results to real world applications may not be possible.

Proper fluid simulation is complex and computationally expensive, but efficient, stable methods have been available to us for some time. In 1999, Jos Stam presented a fluid solver based on a stable solution of the incompressible Navier-Stokes equations (Stam, 1999). These equations describe the flow of incompressible viscid fluids, water being the prime example. Some source code was released, but it was intended as a quite limited proof of concept rather than as a tool for further scientific research in fluid simulation.

Potential flow is inviscid and irrotational, whereas real flows as described by the Navier-Stokes equations can be viscid and rotational, such as those generated by water. Viscosity describes the fluid's resistance to flow, and the rotationality allows for vorticity, the rotation of the fluid around a point. In addition to this true viscosity, there is also numerical viscosity, an untoward effect of numerical approximations in discretisation of the results of the equations in simulations.

A Stam-type solver uses a grid-based method. Multiple grids represent the different aspects of the flow, such as separate grids for density, velocity, and pressure. The boundaries of the grid can be open or closed, allowing the fluid to exit the area or not. Particle-based methods are also used, in which the flow is described by a number of particles, which may encode velocity information or even vorticity information.

Recently, open source tools specifically intended for scientific research into computer graphics fluid simulations have been developed, one such example being mantaflow (Thuerey and Pfaff, 2017), which is used for the generation of data in this study. Tools such as these allow for the user to flexibly set up new scenes in a variety of ways. Multiple kinds of fluid simulation methods are supported, and dimensionality and resolution can easily be changed.

The main goal of this thesis is to train a number of different neural network architecturers to perform the source detection task on data generated by more realistic fluid simulations to determine how well they perform with regards to those trained on potential flow data (Wolf and van Netten, 2018); (Boulogne et al., 2017). Many aspects of the methods are modelled after the two previously mentioned papers, such that a proper comparison can be made between the two sets of results. Preliminary results on this more realistic data should be a better indicator of possible real-life performance of artificial lateral lines used in conjunction with such neural networks on real flows due to their greater similarity.

2 Methods

In the first section, I outline how the scene is set up in mantaflow to generate the data that are used for the training of the neural networks.

In the second section, I discuss how the three neural networks used are evaluated and how they are set up. These three network types are the Multilayer Perceptron (MLP) (Rumelhart and Geoffrey E. Hinton, 1985), Extreme Learning Machine (ELM) (Guang-Bin Huang and Siew, 2006), and Echo State Network (ESN) (Jaeger, 2002).

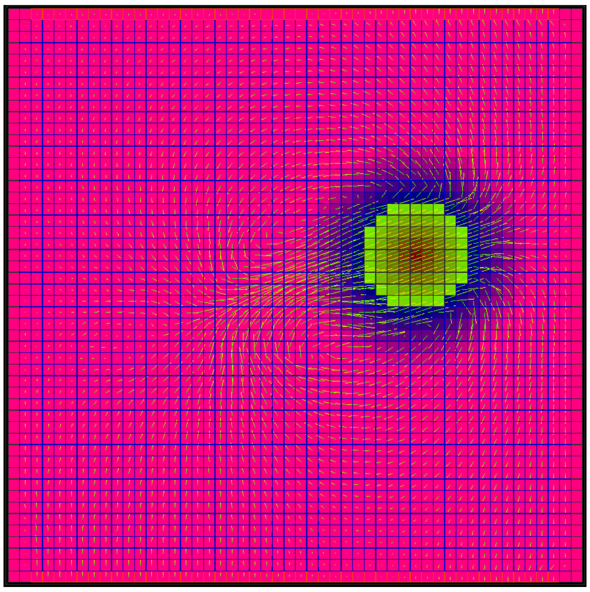
Figure 2.1: An example screenshot of a velocity field produced by the movement of a sphere in mantaflow, highlighting the sphere.

2.1 Data Generation

The data used to train the networks was generated using the open-source framework mantaflow (v0.11), which is intended for fluid simulation research in computer graphics (Thuerey and Pfaff, 2017).

Mantaflow is used to generate 2500 paths of 100 time steps of an object moving randomly through a fluid resembling water at a constant speed. This uses a Stam-type solver, which is a grid-based method, rather than particle-based. Paths are ended after 100 time steps to reset the environment.

For the MLP, 2000 paths are used for training, 250 are used for validation and 250 for testing. For the ELM and ESN, 2000 paths are used for training and the remaining 500 are used for testing.

As opposed to potential flow, the velocities at a certain point away from the source cannot be calculated instantaneously, and are instead calculated iteratively, depending on the previous time steps

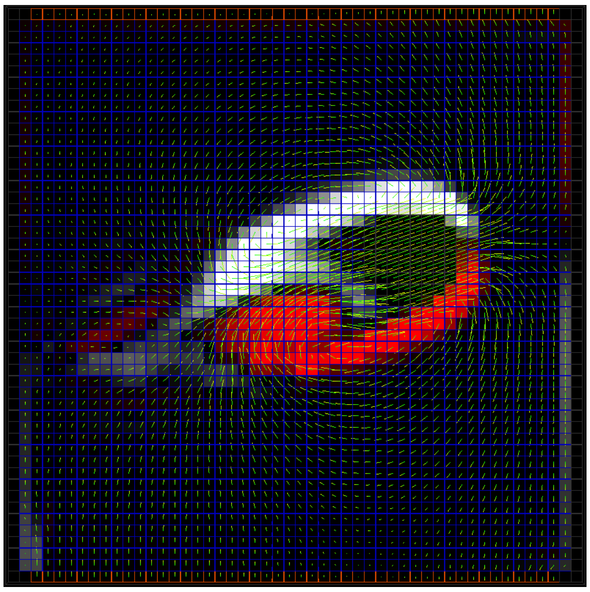

Figure 2.2: This image shows the vorticity generated by the velocity field shown in Figure 2.2. Negative rotation is shown in red, positive rotation in white. The green bars show the velocity vectors.

and the values of surrounding grid cells. Instead, the generated x-y fluid velocities along with the object's x and y positions and angle are written to a separate file for each path after each time step. The sampled velocity profiles serve as the inputs for the neural networks, and the object's position and angle as outputs.

2.1.1 Scene Setup

The scene has a number of properties that need to be set for both the environment and the object itself.

The object is defined as a sphere with a radius, $r = 0.06$ m, moving at a constant $v = 0.13$ m/s in the x,y plane, with a certain angle φ to the x-axis. The angle with which the object travels is changed over time, which gives rise to a random trajectory through the plane (Wolf and van Netten, 2018).

The environment is defined as a 100 cm by 100 cm by 1 cm grid, in which a voxel corresponds to 1 cm^3 of fluid. The size of 1 in the z-plane is necessary, as mantaflow treats a two dimensional simulation as a special case of a three dimensional one, namely one for which the resolution of the z-dimension is 1. This also allows for straightforward extension from two dimensions to three.

All of the boundaries of the grid are set to be open, meaning that the flow is able to exit the grid, as opposed to being closed, in which case the fluid velocities at the edges would be set to zero and compensated for accordingly.

The points at which to place artificial sensors are decided afterwards to be 32 locations centered around $x = 50$ cm, with 2 cm between each sensor, along the bottom side of the grid at $y = 1$ cm. These 64 measurements represent the 2D velocity profile measured by the simulated artificial lateral line for a certain time step.

The kinematic viscosity is set to approximate that of water, such that $\nu = 10^{-6}$ m^{2}s^{-1}. Viscosity is one of the characteristics not present in the potential flow model, and its effects on the measured velocity profiles are profound, see Appendix A for comparisons of velocity profiles with different values for the kinematic viscosity and those generated by a potential flow model.

2.1.2 Path Generation

At the start of each run, the object is placed at a random position in the grid at a random angle. Every five time steps, another random angle between -57.3° and 57.3° (between -1 and 1 radians) is added to the current angle in order to make the object move randomly through the area.

At each time step, a check is performed to see whether the object is about to exit the area, in which case the angle is altered slightly such that it stays within the area. This does, however, cause erratic behaviour at the edges of the grid, and especially in the corners, as those have two edges. The objects are turned away from the edge, but may instantly return, causing the object to hug the sides of the area. See the heatmap in Figure 2.3. The edges and corners have a greater occurrence than the rest of the grid (Wolf and van Netten, 2018).

2.1.3 Data Preprocessing

The recorded x and y positions for the outputs as recorded are scaled from the range [0, 100] to [0, 1] instead.

The angle is recorded in radians and is always kept in [0, 2π]. Once the angle exceeds 2π, it re-

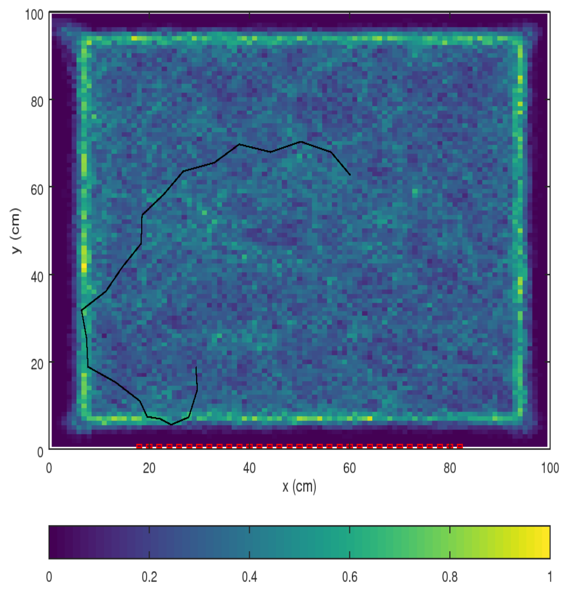

Figure 2.3: A heatmap of all the locations visited by the moving sphere during training. The line shows an example of a path traversed by the sphere. The red boxes along the bottom show the positions of the sensors.

turns to the other side of the range, which causes a problem in calculating the error if the angles are spread between the two extremes of the range. This problem is solved by taking the sine and cosine of the angle as outputs instead; this solves the wrap-around problem, and scales the angle, such that both outputs are in [-1, 1] (Wolf and van Netten, 2018).

The velocity profile inputs are preprocessed in two ways. First, random noise of the level 10^{-6} from a uniform distribution is added to the velocity profiles. This noisy velocity profile is then normalised by dividing each element by the absolute maximum of the velocity profile, so that all elements are in [-1, 1].

2.2 Neural Networks

Each of the three neural network architectures is evaluated by the same five error measures. The first is the average mean squared error (MSE) over all four outputs, which the networks minimise during learning. The average MSE is used as an overview of the performance of the network over all four outputs.

This average MSE is then further split up into the next two measures to give a better overview of the performance on the two subtasks, location estimation and angle estimation. The second measure is the location MSE, which represents the MSE error over the first two outputs, the x and y locations of the object. The third measure is the angle MSE, which represents the MSE error over the last two outputs, the cosine and sine of the angle.

The last two measures are more intuitive versions of the previous two measures. The fourth being the location mean Euclidean distance (MED), which shows the absolute distance between the estimated location of the source and the actual location. The fifth measure is the angle error, in degrees, determined by the last two outputs.

2.2.1 Multilayer Perceptron

The MLP is fully connected, with an input layer of 64+1 nodes, the x and y velocity profiles for the 32 sensors, a single hidden layer with 400+1 nodes, and an output layer of 4 nodes, the x and y location of the source, and the angle of the source, as $\cos\varphi$ and $\sin\varphi$.

The activation function for the hidden nodes is $f(x) = \tanh x$. The activation function for the output nodes is the identity function, $f(x) = x$. Bias nodes are added on the input layer as well as on the hidden layer, represented by the +1 above, whose activation is always one. All weights are initialised uniformly at random between $[-\frac{1}{2}, \frac{1}{2}]$.

The network is trained using the backpropagation algorithm with a learning rate $\eta = 10^{-2}$ over 100 epochs. The error over a validation set is monitored to determine whether overfitting takes place during training.

2.2.2 Extreme Learning Machine

The ELM used has a hidden layer of 400 nodes. Its input layer, output layer, and the activation functions are identical to the MLP. This number of hidden nodes was determined to be optimal by Wolf (2018). The ELM also has a bias node on the input layer and on the hidden layer.

The weights from the input layer to the hidden layer are also fully connected, and the weights are also initialised uniformly at random between

$[-\frac{1}{2}, \frac{1}{2}]$. These weights are not adjusted further. The network is trained by determining the weights from the hidden layer to the output layer by using the pseudoinverse of H to calculate W, such that $W = TH^{\dagger}$, where W is the weights matrix, T is the goal output matrix, and H is the hidden activation matrix (Guang-Bin Huang and Siew, 2006).

2.2.3 Echo State Network

An ESN consists of an input layer, a dynamic reservoir (hidden layer), and an output layer. The input layer and output layer are both fully connected to the dynamic reservoir, but the dynamic reservoir also has sparse connections with itself. The ESN used has a sparcity of 0.2, a spectral radius of 0.1, and a dynamic reservoir of 400, matching the ELM, as well as a bias node on the input layer and on the hidden layer.

The first 50 outputs of the ESN are discarded to cancel out the effects of the ESN's starting state, because of their inaccuracy caused by the short term memory of the network. This number is also known as the washout time of an ESN (Jaeger, 2002).

3 Results

The performance of all three networks in both environments for all measures is shown in table 3.1. A box plot of one of the intuitive measures, the error in mean Euclidean distance is shown in figure 3.1. A box plot of the other measure, the angle error in degrees, is shown in figure 3.2.

In these box plots, the whiskers start at the first vigintile, and end at the nineteenth vigintile. The boxes themselves range from the first quartile to the third quartile. The line marks the median, the diamond the mean. The other points plotted at the far ends show the hundredth percentile in the Euclidean distance box plots, and the zeroth and hundredth in the angle error box plots.

In nearly each case, the networks perform slightly better on their training sets than on their test sets, which is to be expected. The MLP, however, performs better on the test set with regards to location estimation, but this is not a reason to suspect overfitting. The difference is most likely a result of the smaller size of the test set. The difference in performance therefore appears to be small enough to assume no overfitting has taken place.

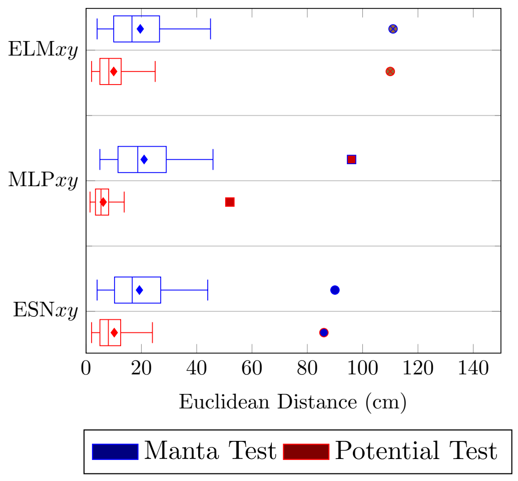

Figure 3.1: A box plot showing the quartiles of the mantaflow and potential flow test (N = 50000 for ELM and ESN, N = 25000 for MLP) results in terms of location estimation, xy. The diamond shows the mean of the errors, while the extra markers plotted at the extremes indicate the hundredth percentile.

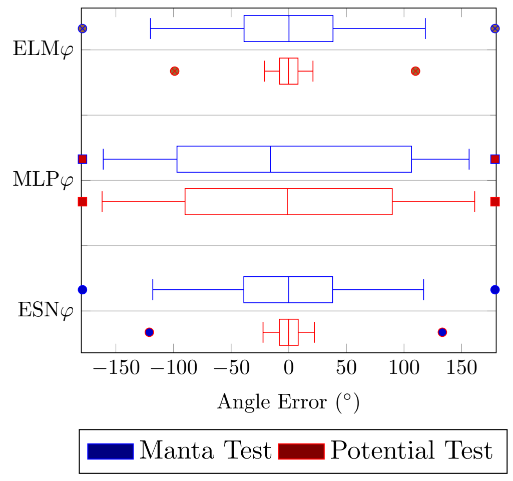

Figure 3.2: A box plot showing the quartiles of the mantaflow and potential flow test (N = 50000 for ELM and ESN, N = 25000 for MLP) results in terms of angle estimation, φ. The diamond shows the mean of the errors, while the extra markers plotted at the extremes indicate the zeroth and hundredth percentiles.

Table 3.1: Table of performance of ELM, MLP, and ESN in the potential flow and mantaflow environment after training on train and test sets with added noise level 10^{-6} m/s.

Network	Error	Potential flow		Mantaflow	
		Train	Test	Train	Test
ELM_{400}	Average MSE	0.025	0.026	0.187	0.192
	Location MSE	0.008	0.008	0.027	0.026
	Angle MSE	0.043	0.044	0.348	0.356
	Mean Euclidean Distance (cm)	10.1	10.2	19.4	19.4
	Average Absolute Angle Error (°)	10.4	10.5	49.5	50.2
MLP_{400}	Average MSE	0.032	0.007	0.740	0.920
	Location MSE	0.012	0.006	0.058	0.062
	Angle MSE	0.010	0.038	0.681	0.731
	Mean Euclidean Distance (cm)	6.8	6.2	20.7	21.3
	Average Absolute Angle Error (°)	89.9	90.1	88.8	95.4
ESN_{400}	Average MSE	0.025	0.026	0.186	0.191
	Location MSE	0.008	0.008	0.026	0.027
	Angle MSE	0.042	0.043	0.346	0.356
	Mean Euclidean Distance (cm)	10.1	10.2	19.3	19.4
	Average Absolute Angle Error (°)	10.3	10.3	49.2	50.4

3.1 Location Estimation

The mantaflow results are slightly worse than the potential flow results in every case. The average distance between the estimated source location and the actual source location is about 20 cm, twice as large as the error for the potential flow data results on the ELM and ESN, which is around 10 cm. The MLP performs better than the ESN and ELM on the potential flow data, around 6 cm versus 10 cm, but roughly matches their performance on the mantaflow data so that its performance decreases with factor of three. The location estimation performance on the mantaflow data is roughly the same for each network type, despite a discrepancy in performance on the potential flow data.

3.2 Angle Estimation

The average angle estimation error also suffers under the more realistic data. For the potential flow data, it is on average about 10° for the ELM and ESN and 90° for the MLP. The mantaflow results are nearly the same for the MLP, 90°, but several times worse for the ELM and ESN, namely around 50°.

3.3 Transfer Learning

A network was also trained on potential flow data, but then tested on the mantaflow data. Those results are in table 3.2. Corresponding box plots can be found in figures 3.3 and 3.4.

In this case, the location estimation error suffers greatly on the training set, for which the network has a location estimation error of around 10 cm, which increases to 78 cm on the mantaflow data. Likewise, the angle estimation over the potential flow data is around 10°, which increases to 91° on the mantaflow data.

A network trained on potential flow data and tested on mantaflow data performs about four times as badly on location estimation and around twice as badly on angle estimation.

4 Discussion

In the introduction, I set out to recreate results of neural networks trained to perform source localisation, i.e. location estimation and angle estimation, on a sphere moving through a plane.

I then described in the methods section how the realistic data were generated using an open source fluid simulation framework called mantaflow, and

Table 3.2: Table of performance of an ELM$_{400}$ trained on potential flow data, but tested on mantaflow data.

Error	Potential flow Train	Potential flow Test	Mantaflow Test
Average MSE	0.026	0.026	1.095
location MSE	0.009	0.009	0.418
Angle MSE	0.043	0.044	1.772
Location MED (cm)	10.4	10.5	78.0
Average Angle Error (°)	10.269	10.373	91.04

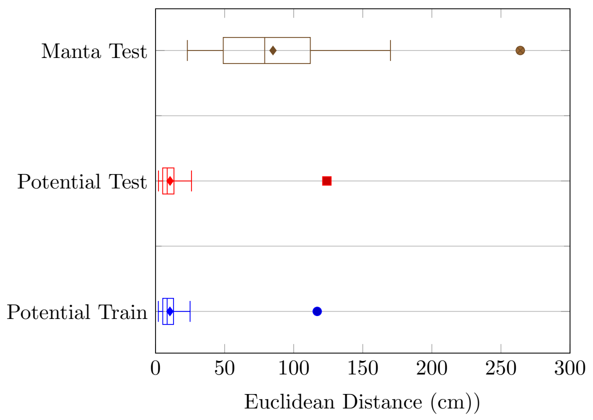

Figure 3.3: A box plot of performance for an ELM$_{400}$ trained (N = 200000) and tested (N = 50000) on potential flow data, but also tested (N = 50000) on mantaflow data in terms of angle estimation error in degrees.

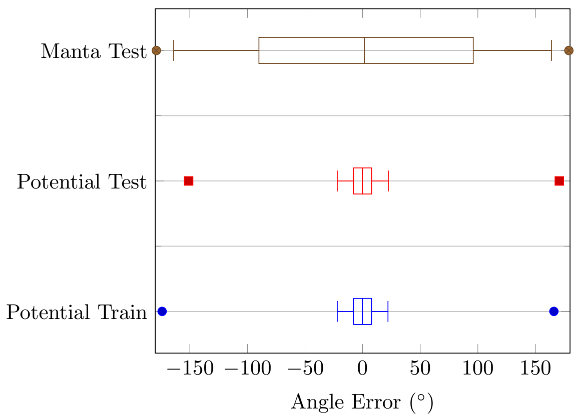

Figure 3.4: A box plot of performance for an ELM$_{400}$ trained (N = 200000) and tested (N = 50000) on potential flow data, but also tested (N = 50000) on mantaflow data in terms of angle estimation error in degrees.

how three types of artificial neural network, MLP, ELM, and ESN, that were used were set up.

In the results, I presented the performance of the different neural networks on both mantaflow data and potential flow data. In every case, the mantaflow data proved to be significantly more challenging for the networks than the potential flow data.

The average angle error increases nearly five fold, which is worse than the decrease in performance of location estimation. These errors seem to be normally distributed in all cases. The spread of these errors on the mantaflow data, however, is much greater than on the potential flow data.

Taking into account that the radius of the sphere is about 6cm, the location error in the potential flow case might be considered to be only about 4cm or 14cm in the mantaflow case. A location estimated past the radius of the object completely misses the object, and so the location error could be said to be roughly three and a half times worse instead.

From the box plots we can tell that the distribution of errors with regards to location estimation is positively skewed. This means that most of the errors fall in the lower range of the distribution. This does not excuse the much greater number of errors in the higher ranges compared to the potential flow case, but does show that there are more good performance cases than bad. Isolating these bad performance cases and studying them may tell us more about how or why the occur. We could then start to think about how to treat these cases in order to increase performance.

I also showed the results of training a network on potential flow data, and then testing on mantaflow data. Comparing the test results from the potential flow test data and the mantaflow test data indicates that it transfer learning isn't possible. A neural network trained on potential flow data will most likely perform poorly in a real world setting.

In conclusion, the networks trained on mantaflow suffered a slight decrease in performance over all measures when compared to their performance on potential flow data, despite the increased complexity of the data. The angle estimation performance decreased by a factor of five. The location estimation performance on average only suffered a decrease in performance of around a factor of two, which could be considered acceptable, taking into account the increased complexity of the problem.

4.1 Limitations

The networks used on the mantaflow data had hyperparameters identical to those used for the potential flow data. No additional tuning was performed on the mantaflow data, which could have been detrimental to its performance, due to the possible greater complexity of the flows. There were two distinct causes behind this. First, using the exact same ANNs seemed to give a more apt comparison between the performance on the two different kinds of data. Second, due to constraints of time, and difficulties with the data generation at the start of the project, much more time was spent on the data generation aspect, rather than on the training and testing of the ANNs.

The path generation algorithm employed here seems to enjoy a fondness for edges and corners over the rest of the grid. This might adversely affect the results due to overrepresentation of velocity profiles that are caused by weird movements at the edges of the grid that are a result of the sphere being forced not to exit the area. An increase in performance may be achieved through better and more equal coverage of the entire area of interest.

We attempted to approximate the properties of water in this simulation. A comparison between empirically measured water flow velocities as measured by real sensors and the flows measured by the simulated sensors in simulation was planned, but, due to time constraints, never performed. Depending on the results of this comparison, the above results may be confirmed to hold water or not.

The movement of the object is also quite unnatural. It moves at a perfectly constant speed, and changes direction maximally about 57° instantaneously. It may have been better to ease the object in this change of angle or to change the speed of the object.

The MLP did not seem to learn how to estimate the angle properly, as the performance on potential flow and mantaflow data is roughly the same. This may have led to increased performance on location estimation on the potential flow data, but the same does not hold for the mantaflow data. More importantly, the MLP still showed a decreasing error in both the validation set and the training set when training was stopped. It would appear that the MLP had not yet reached an optimum, and training it for longer may lead to better results.

4.2 Future Work

In the future, it may be beneficial to work on some of the points raised above.

An attempt to tune the hyperparameters of the networks to perform better on the mantaflow data might generate better results than simply reusing the hyperparameters used for potential flow data, taking into account the increased complexity of the problem. Other considerations to increase the performance such as different data preprocessing techniques, and different neural network architectures could also be made.

Another consideration could be looking into a better path generation algorithm, that more evenly distributes itself across the entire space, rather than being tricked into hugging the walls and corners repeatedly.

In an effort to help with development of such systems for use in the real world, it would be incredibly helpful to be able to tune the simulation environment perfectly to the qualities of the actual environment in which it will be deployed. This can be done by comparing the fluid velocity profiles of real world flows with simulated flows, and tweaking the latter until they match up almost completely. This will hopefully allow for transfer of a system trained on simulated data to the real world, bypassing the difficulties of generating large amounts of data in the real world on which to train the system.

In this study, the sensor array was placed along only one axis, at equidistant locations. To test the performance of the networks on different, perhaps more optimal, sensor array configurations might increase the performance on source localisation.

Additionally, this project and previous work has focused solely on two dimensional data. It may be interesting to explore three dimensional simulations as well, taking into consideration fluid velocities in the z-direction. This 2D to 3D conversion could be achieved easily by changing a handful of parameters in the mantaflow scene setup.

References

Luuk H Boulogne, Ben J Wolf, Marco A Wiering, and Sietse M van Netten. Performance of neural networks for localizing moving objects with an artificial lateral line. *Bioinspiration & Biomimetics*, 12(5):056009, 2017.

Branislava Curcic-Blake and Sietse M. van Netten. Source location encoding in the fish lateral line canal. *The Journal of Experimental Biology*, 209: 1548–1559, 2006.

Sven Dijkgraaf. The functioning and significance of the lateral-line organs. *Biological Review*, 38(1): 51–105, 1963.

Qin-Yu Zhu Guang-Bin Huang and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach., 2002.

David E. Rumelhart and Ronald J. Williams Geoffrey E. Hinton. Learning internal representations by error propagation. Technical Report DTIC Document, 1985.

Jos Stam. Stable fluids. In *SIGGRAPH 99 Conference Proceedings, Annual Conference Series*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

Nils Thuerey and Tobias Pfaff. MantaFlow, 2017. *http://mantaflow.com.*

Ben J Wolf and Sietse M van Netten. Training submerged source detection for a 2d fluid flow sensor array with extreme learning machines. 2018.

A Appendix

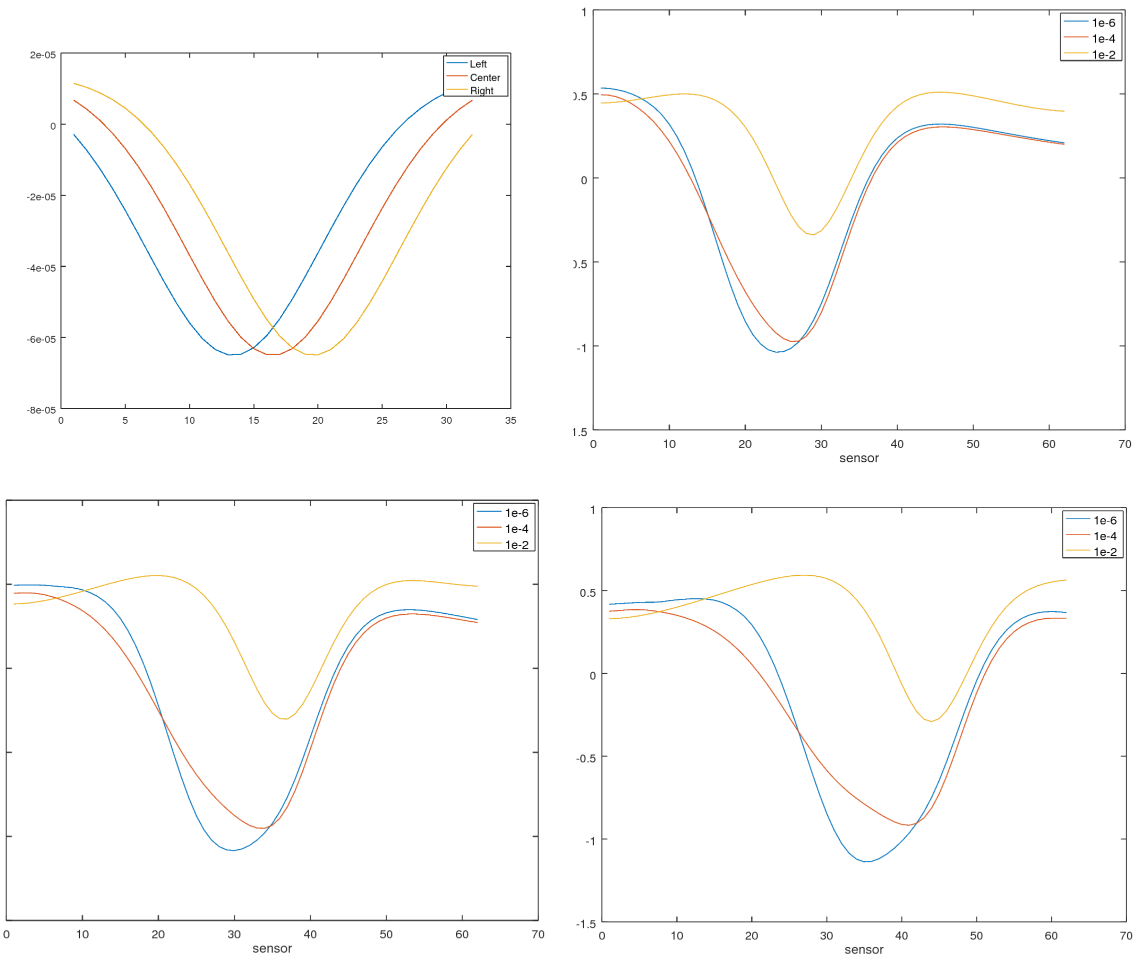

Figure A.1: The top left graph shows potential flow x velocity profiles at different x positions. The other three show x velocity profiles for a linearly moving sphere at the same x positions for 3 different kinematic viscosity units. Top right, x = 0.3; bottom left x = 0.5; bottom right x = 0.7, corresponding to Left, Center, and Right in the first plot.

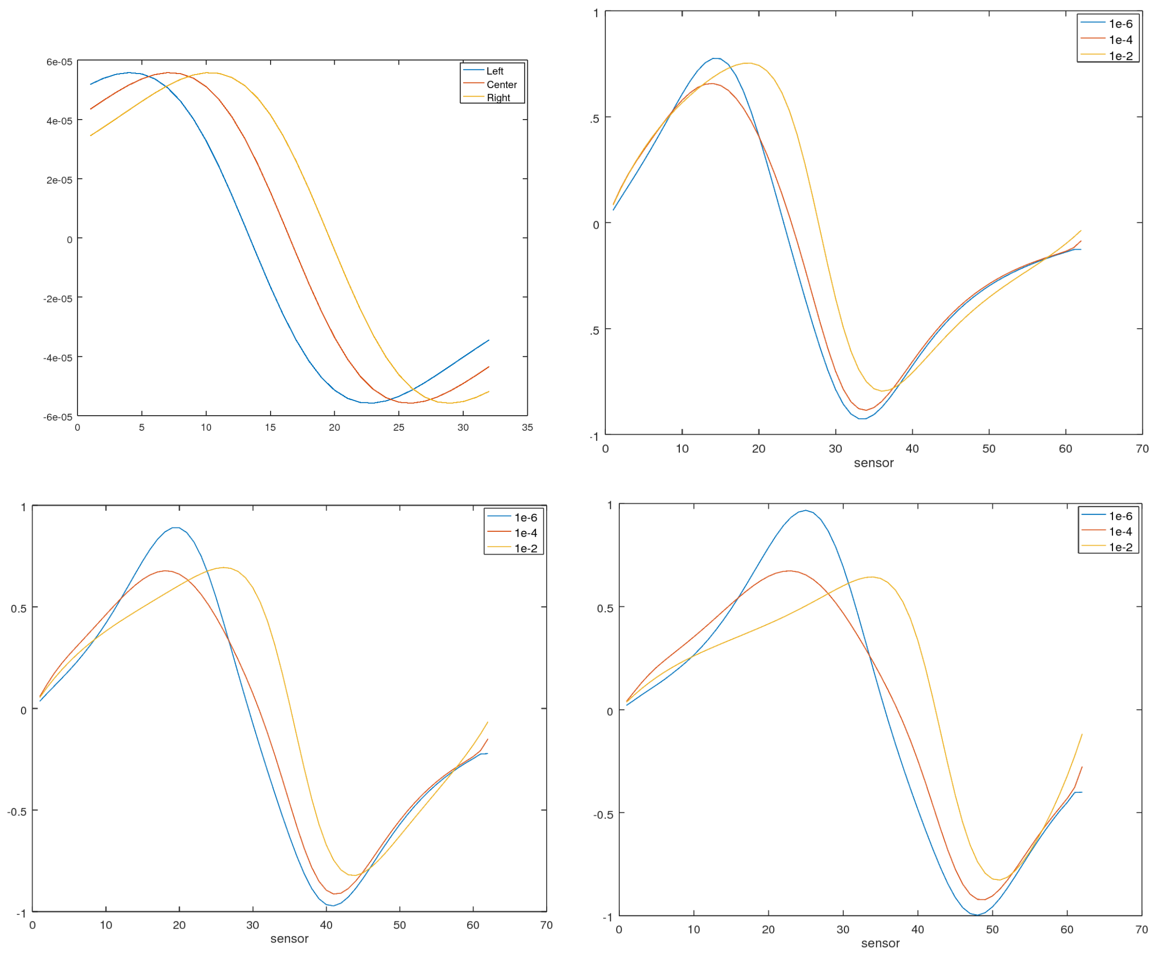

Figure A.2: The top left graph shows potential flow y velocity profiles at different x positions. The other three show y velocity profiles for a linearly moving sphere at the same x positions for 3 different kinematic viscosity units. Top right, x = 0.3; bottom left x = 0.5; bottom right x = 0.7, corresponding to Left, Center, and Right in the first plot.