Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Eirik Søvik

Systematic Analysis of Experiments on Sub-Carangiform Fish Hydrodynamics

Master's thesis in Marine Technology
Supervisor: Marinela Greco
May 2022

NTNU
Kunnskap for en bedre verden

Eirik Søvik

Systematic Analysis of Experiments on Sub-Carangiform Fish Hydrodynamics

Master's thesis in Marine Technology
Supervisor: Marinela Greco
May 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Kunnskap for en bedre verden

MASTER THESIS IN MARINE TECHNOLOG

Spring 2022

FOR

Eirik Ruben Grimholt Søvik

Systematic Analysis of Experiments on Sub-carangiform Fish Hydrodynamics

(Systematisk analyse av eksperimenter på subkarangiform fiskhydrodynamikk)

Fish can be quite fast and efficient swimmers. This has motivated great research effort to better understand the unsteady fish hydrodynamics relevant for these fish skills to transfer them into novel engineered marine vehicles. A previous master thesis has investigated one-speed case of tests on sea-bass locomotion carried out in an authorized CNR-IAS lab, within a collaboration of the Italian research institute CNR-INM with the Centre of Excellence NTNU AMOS. The study was based on video-image analysis and use of Fast Fourier Transform (FFT) and of Least-Square-Method (LSM) with attempt to identify the fish motion, its envelope, its connected frequencies, wavelengths, and the law of the wave motion of the fish. The proposed approaches were promising but the estimates based on them presented differences with respect to the actual fish-locomotion properties.

During the project work prior the present master thesis, the student selected and described Machine Learning algorithms (MLA) to be used as alternative approaches for describing and analyzing subcarangiform fish locomotion. Another experimental activity on salmon locomotion available from SeaLab in Trondheim were also examined to be used in the MSc analysis.

Objective

The present master thesis aims to use the selected MLAs for a systematic fish-hydrodynamic analysis for subcarangiform fish locomotion at different swimming speeds using available experimental data.

The work should be carried out in steps as follows:

1. Summarize major findings/outcomes from the project thesis, reporting on the background motivation, literature study, description of the selected MLA modelling and of the analysis so far carried out; possibly complement the literature survey in order to identify state-of-the-art of the problem.
2. Describe the experiments on fish free swimming run in a swim tunnel available from SeaLab. Examine the conditions for the tests, as well as the measurements done and select the cases to be investigated. Identify and discuss possible error sources/limitations in the experiments.
3. Create a working environment in python using MLAs. Implement the selected MLAs and assess their reliability considering idealized, pre-set (i.e. known) fish swimming scenarios. Apply the MLAs to characterize the fish locomotion for the experiment described in step 2 by means of video-image analysis and combining with other tools (e.g., FFT) within a suitable analysis strategy.
4. As first attempt of validation of the methodology proposed in step 3, compare the identified fish features in terms of law of the wave motion of the fish, involved oscillation frequencies and wavelength as a function of the swimming speed with those available for sub-carangiform fish swimming from the literature study in step 1.

5. Draw the conclusions from the studies carried out and their results and discuss possible further research steps.

The work may show to be more extensive than anticipated. Some topics may therefore be left out after discussion with the supervisor without any negative influence on the grading.

The candidate should in his report give a personal contribution to the solution of the problem formulated in this text. All assumptions and conclusions must be supported by mathematical models and/or references to physical effects in a logical manner.

The candidate should apply all available sources to find relevant literature and information on the actual problem.

The thesis should be organised in a rational manner to give a clear presentation of the work in terms of exposition of results, assessments, and conclusions. It is important that the text is well written and that tables and figures are used to support the verbal presentation. The thesis should be complete, but still as short as possible. In particular, the text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The thesis must contain the following elements: the text defining the scope (i.e. this text), preface (outlining project-work steps and acknowledgements), abstract (providing the summary), table of contents, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables and equations shall be numerated.

The supervisor may require that the candidate, in an early stage of the work, present a written plan for the completion of the work. The plan should include budget for the use of computer and laboratory resources that will be charged to the department. Overruns shall be reported to the supervisor.

From the thesis it should be possible to identify the work carried out by the candidate and what has been found in the available literature. It is important to give references to the original source for theories and experimental results.

Supervisor : Marilena Greco
Co-supervisor : Claudio Lugni
Co-supervisor : Annette Stahl

Submitted : January 15th 2020
Deadline : May 23rd 2022

Marilena Greco
Supervisor

Preface

This is the culminating work of a integrated masters program at the department of Marin Technology at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The work of this thesis was started in the spring of 2021, but were postponed and finished in the spring of 2022, due to personal matters.

The topic of this master's thesis is that of fish locomotion, which is a cutting edge field of study in hydrodynamics, as well as robotics and other related fields. The advent of new scientific knowledge and methods in addition to technological advancements mean that bio-inspired robotics is an up and coming field of research. The possibilities are almost endless. This is not news to Marilena Greco, who is the lead supervisor of this thesis and who works at the Centre of Excellence NTNU AMOS. In addition I have recieved much help from Adjunct Professor Claudio Lugni at the Italian research institute CNR-INM, but who is also employed at the department of Marine Technology.

Knowledge about hydrodynamics, fish locomotion or machine learning is not needed to read this thesis, although they will increase comprehension.

Acknowledgments

I must first thank to my main supervisor Professor Marilena Greco for having been so supportive during a difficult time, and for having been so helpful all the way through. Our weekly meetings who have sometimes been short and swift, and other times dragged on into detailed discusisons on various implementations have always been helpful and uplifting. I must also thank Adjunct Professor Claudio Lugni for having provided me with much needed help and insight when I could not find a way through myself.

Thanks to Huili Xu for providing me with so good experimental data, and to Anette Stahl for providing me with a challenging, but rewarding deep learning framework to work on. I also want to mention Halvard Kværna, whos help was essential in solving the troubles encountered early on with both linux, python and detectron2. Without you, we wouldn't even have gotten started.

The knowledge and expertise that Greco and Lugni have shown truly showcase how the department of Marine Technology has been instrumental in building up Norways top-of-the-world expertiece in all things offshore. I am especially exited to see what the future will hold for them and bio-inspired hydrodynamics.

Abstract

The goal of this thesis is to examine if there is a way to systematically gather data from fish swimming experiments so that future data extraction can be much faster.

A precoursing work has been done to map out this possibility, which concluded that it should indeed be possible, and indicated that detectron2 would be a machine learning algorithm to base the work off of.

The purpose of this thesis is then to investigate a systematic approach to fish locomotion analysis using machine learning object identification and segmentation to extract fish outlines from video experiments of swimming fish. This is done using detectron2, a state of the art deep learning framework developed by FacebookAI, for object detection and segmentation. Segmentation is when the outline of objects are found, which is essential to what this thesis sets out to accomplish. Once these outlines, or masks, are found, they are processed so that they are usable for data extraction of the fish midline. The fish midline is the spine of the fish, or in other words the middle line.

Once the masks are produced, they are handled by another program, a midline extractor. This function extracts them by a simple algorithm which is not robust to other test setups. A much more robust method of extraction was tried to be implemented, but it was not possible to achieve usable results.

Lastly, the resulting midlines were processed in a simple hydrodynamical analysis to check that they not only looked good in plots, but that the methods described above actually produce results that can be used for research.

The thesis has been deemed a success, due to time-concerns a deeper hydrodynamical analysis could not be performed, but the main goals of the thesis were still achieved.

Sammendrag

Målet med denne oppgaven er å undersøke om det er en måte å systematisk samle data fra fiskesvømmeeksperimenter slik at fremtidig datautvinning for påfølgende analyse kan foregå mye raskere.

Det er gjort et forarbeid for å kartlegge denne muligheten, som konkluderte med at det faktisk burde være mulig, og indikerte at detectron2 ville være en maskinlæringsalgoritme å basere arbeidet på.

Hensikten med denne oppgaven er deretter å undersøke en systematisk tilnærming til fiskebevegelsesanalyse ved bruk av maskinlæringsobjektidentifikasjon og segmentering for å trekke ut fiskekonturer fra videoeksperimenter av svømmende fisk. Dette gjøres ved hjelp av detectron2, et toppmoderne rammeverk for dyplæring utviklet av FacebookAI, for objektdeteksjon og segmentering. Segmentering er når omrisset av objekter blir funnet, noe som er avgjørende for hva denne oppgaven skal oppnå. Når disse konturene, eller maskene, er funnet, blir de behandlet slik at de kan brukes til dataekstraksjon av fiskens midtlinje. Fiskens midtlinje er ryggraden til fisken.

Når maskene er produsert, håndteres de av et annet program, en midtlinjeekstraktor. Denne funksjonen trekker dem ut med en enkel algoritme som ikke er robust for andre testoppsett. En mye mer robust metode for utvinning ble forsøkt implementert, men det var ikke mulig å oppnå brukbare resultater.

Til slutt ble de resulterende midtlinjene behandlet i en enkel hydrodynamisk analyse for å sjekke at de ikke bare så bra ut i plott, men at metodene beskrevet ovenfor faktisk gir resultater som kan brukes til forskning.

Oppgaven har blitt ansett som en suksess, på grunn av tidsmessige bekymringer kunne en dypere hydrodynamisk analyse ikke utføres, men hovedmålene for oppgaven ble likevel oppnådd.

Contents

List of Figures

List of Tables

Nomenclature

BL Fish body length

$centroid$ Centre of area of fish outline coordinate

$COCO$ A machine learning dataset

$detectron2$ A deep learning framework

DL Deep learning

fft Fast fourier transform

$MASB$ Medial Axis Shrinking Ball

$masbpy$ Medial axis shrinking ball python implementation

$mask$ segmentation mask provided by machine learning

ML Machine learning

np numpy, python library

$VoTT$ Visual object Tagging Tool

λ Wave length

ω Angular frequency

$B(x,t)$ Recoil term in h(x,t)

$G(x,t)$ Dimensionless amplitude envelope for fish locomotion

$h(x,t)$ Model of fish lateral displacement

k Wave number

s_i Dimensionless envelope parameters

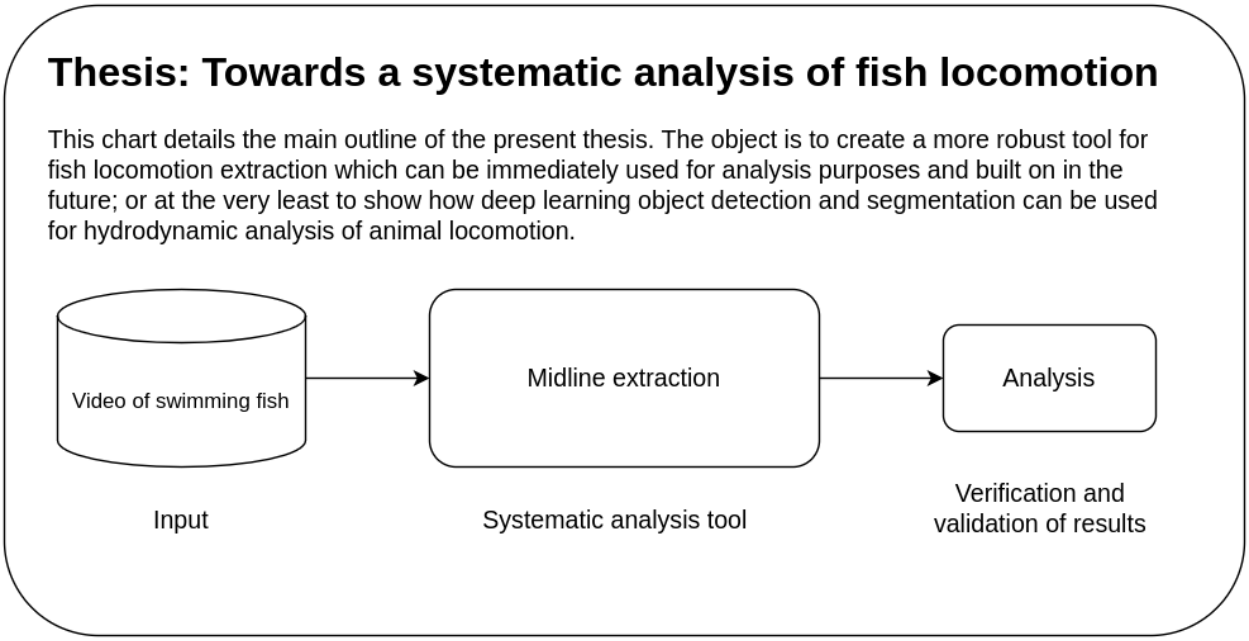


Figure 1: Graphic summarising the present thesis.

1 Introduction

To put it simply, fish move expertly through water, our creations don't. Bio-inspired engineering has existed for a long time, but recent improvements in technology allow us new ways to explore this realm, which up until now has been largely theoretical in nature. The emergence of artificial intelligence, advances in robotics and new ways of analysing and gathering data allow us exiting possibilities, opening the door for new breakthroughs. Even minor improvements in ship propulsion and reduction in drag would yield substantial reductions in global maritime emissions. Alternatively, bio-inspired autonomous robots might be able to do a host of different activities, from upholding sovereignty to monitoring wildlife, to performing search and rescue missions. Already such implementations are being witnessed in aerial drones, marine implementations are a logical next step.

What then, is the objective of this master's thesis? It is to create a tool for systematically analysing fish locomotion quantitatively and then through analysis of the findings verify that the results are usableTODO:phrasing. A summary of the thesis can be found by inspecting figure 1. To accurately account for the relatively unpredictable behaviour of a fish, a deep learning tool is implemented to recognize fish and extract its outline. Then an algorithm for extracting the spine, or the midline, of the fish is used, upon which analysis can be performed.

Also detailed are some methods that were tried but did not yield acceptable results, notably the Medial Axis Shrinking Ball algorithm might prove to be a valuable asset once certain flaws have been addressed. It represents the possibility of extracting midlines from fish swimming in all directions, thus making analysis of complex movement patterns and interactions with other fish and objects possible. The method that was alternatively implemented was a linear method as-

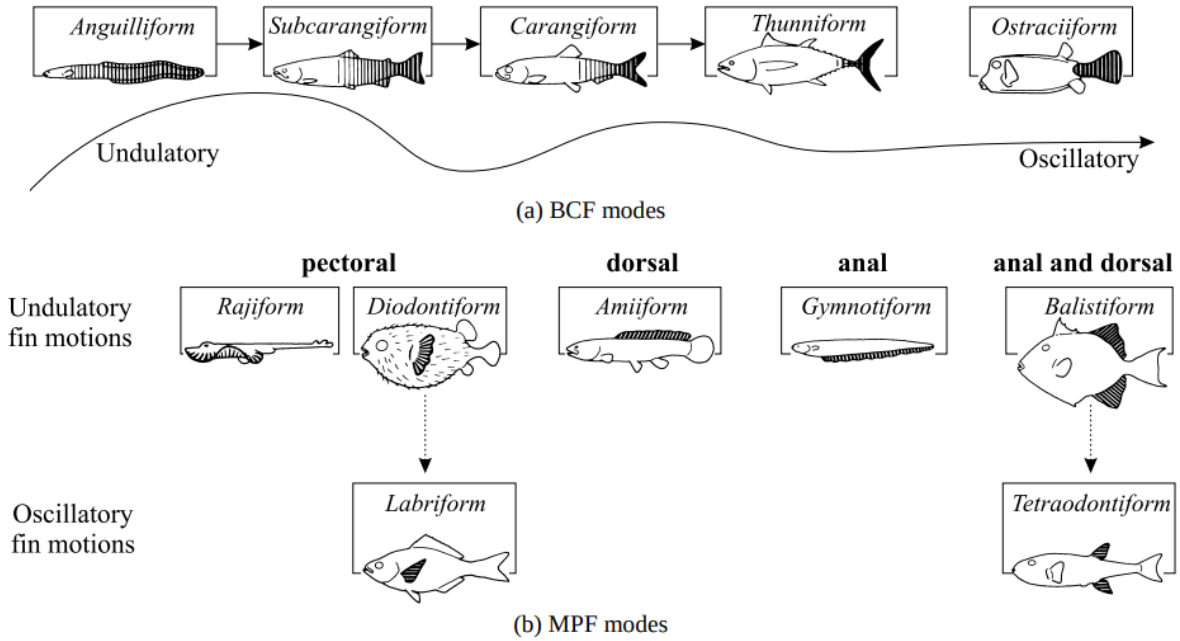


Figure 2: This figure details the basic types of fish locomotion, and shows how the body of the fish is involved in producing undulations. Adopted from (Sfakiotakis et al., 1999)

suming that the fish is oriented along the x-axis, here named the rib approximator.

1.1 Background

1.1.1 Hydrodynamics

Fish locomotion has been a subject of study for a long time, however the nature of this motion has proved to be impossible to emulate, although several physical models have been made. This is due to technological limitations. In this section, some of the theory describing fish locomotion is presented, which will later be used in the analysis of the data found by using the methods devised in this thesis.

The different types of fish locomotion can be classified into several different categories (Breder, 1926), such as listed below in figure 2

Anguilliform movement This is the type of locomotion used by eels and watersnakes, in which the entire body generates undulations which move from the head of the body towards the tail.

Carangiform The majority of the body is used for oscillations, but the head and fore part of the fish does not contribute. They do, however, move as a result of the motions of the rest of the body.

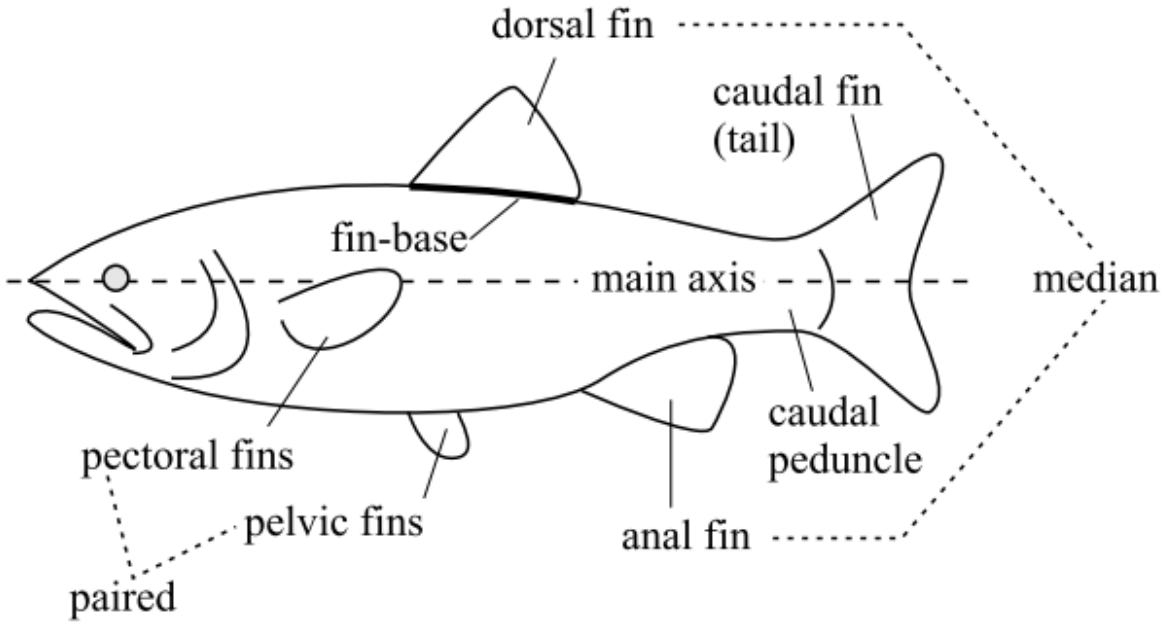


Figure 3: Graphic detailing the basic layout of fish anatomy, adopted from (Sfakiotakis et al., 1999)

Subcarangiform This is the locomotion type that will be examined in this project thesis. A smaller part of the body is used than in carangiform movement, but a majority is still used.

Thunniform Inspired by the locomotion of tuna, in this mode, only the tail-part of the fish is used to generate thrust. This form is specially suited for high speeds.

Ostraciiform movement The type of movement used by sea animals with limited body movement, where the fins are the only part of the body generating thrust.

The subcarangiform mode, which will be analysed in this thesis, uses most of the body, except for the front. This is known as BCF locomotion, or body-caudal-fin locomotion. The front part is stiffer than in anguilliform locomotion, while still allowing for plenty of maneuverability. The basics of fish physiology is described in figure 3. Note especially the caudal fin and caudal peduncle. The main difference between these is that the caudal fin is not composed of any skin or muscle and is much thinner.

The way that a fish generates propulsion is by undulating in the water, as this produces both thrust and a Karman street of vortices. When the fish undulates, it creates vortices in a different way than when a flow encounters an inanimate object such as a cylinder. This change in the von Karman vortex street means that more thrust is generated than in the inanimate case. In addition to this, thrust is generated in the tail as it undulates, creating both a yaw moment on the fish as well as sway and thrust forces. There is therefore a constantly changing flow field related to locomotion.

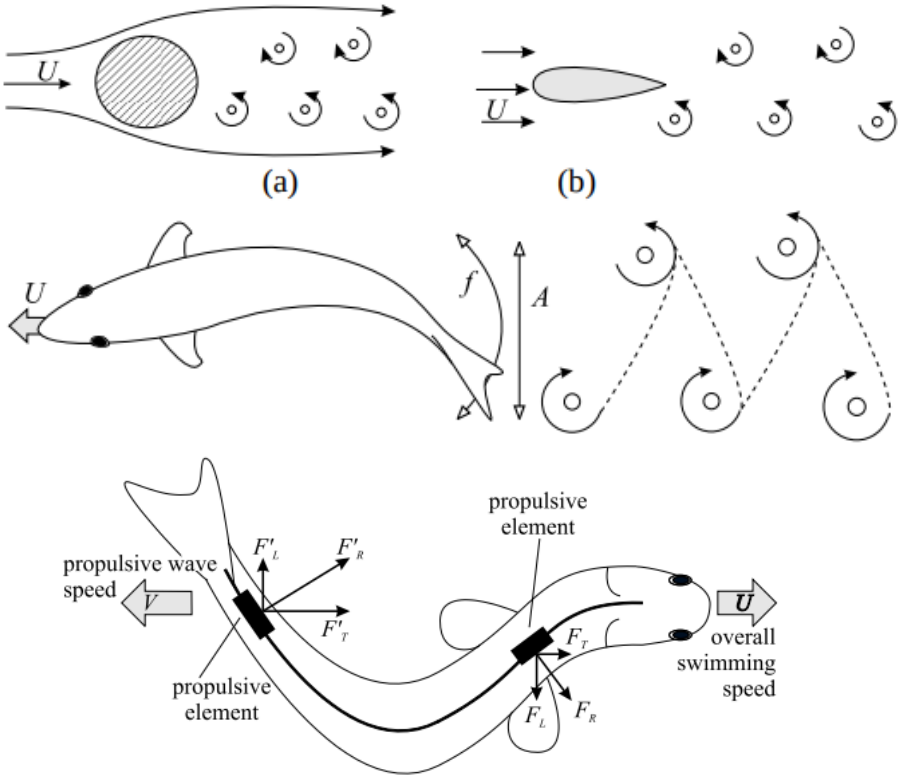


Figure 4: Illustration of how fish generate thrust. (Sfakiotakis et al., 1999)

Mathematical descriptions

There have been many attempts to understand fish locomotion in a theoretical manner, such as citeMaertens,Gao,Triantafyllou, titled "Optimal undulatory swimming for a single fish-like body and for a pair of interacting swimmers", which provides a mathematical description of fish locomotion.

The authors "employ travelling wave kinematics that resemble those observed in fish according to either carangiform og anguilliform swimming, and include recoil" (Maertens et al., 2017). This means that an expression is created containing an expression for the body deformation (h_0), recoil (B), and steering(y_1), given as:

$$\begin{aligned} h(x,t) &= h_0(x,t) + B(x,t) + y_1(x) && (1)\\ &= a_0 A(x) \sin 2\pi(x/\lambda - ft + \phi)) + B(x,t) + y_1(x) && (2)\\ &= g(x) \sin 2\pi(ft + \psi(x))) + y_1(x) && (3) \end{aligned}$$

Here, a_0 is a constant, and $A(x)$ is the envelope of the backwards propogating wave, with wavelength λ and frequency f. Note that $A(1) = 1$.

Recoil term: The recoil

$$B(x,t) = (a_r + b_r x) \sin\left(2\pi(ft + \phi_r)\right))$$

Steering term:

$$y_1(x) = C(x^2 + \gamma x + \beta)$$

Modal analysis of fish locomotion The midlines representing the fish motion can be expressed in both space s and time t as a matrix of dimensions $S \times T$,

$$\boldsymbol{y}_{midline} = \begin{bmatrix} y_{x_1,t_1} & y_{x_1,t_2} & \cdots & y_{x_1,t_T} \\ y_{x_2,t_1} & y_{x_2,t_2} & \cdots & y_{x_2,t_T} \\ \cdots & \cdots & \cdots & \cdots \\ y_{x_S,t_1} & y_{x_S,t_2} & \cdots & y_{x_S,t_T} \end{bmatrix} \quad (4)$$

where $x_s, s\epsilon(1,2,...,S)$ make up the 20 points representing the x-position of the midline and $t_n, n\epsilon(1,2,...,T)$ represents time from the start of the measurments to the end. Note that to simplify calculations only the displacement in the y-direction is accounted for. This means that the results will not be entirely correct, but considering displacement envelopes typically exhibit maximum displacements of around 10% of the fish bodylength, this is a reasonable enough assumption.

Both the samples in space and time need to be uniform. The first objective is to find the traveling index, which is found as

$$\alpha = 1/\text{cond}(\boldsymbol{W}) \tag{5}$$

where $\text{cond}(\boldsymbol{W})$ is the condition number of the complex eigenvectors $\boldsymbol{W}$ of the complex correlation matrix $\boldsymbol{R}$. To find this matrix, we first generate the analytical signal for each midline point, as a function of time,

$$\boldsymbol{z}_s = \boldsymbol{y}_s(t) + iH(\boldsymbol{y}_s(t)) \tag{6}$$

where $\boldsymbol{y}_s(t)$ is the midline point corresponding to x_s and t_n, and $i = \sqrt{-1}$ and $H(\boldsymbol{y}_s(t))$ is the Hermitian transform of $\boldsymbol{y}_s(t)$. This results in a matrix similar to the one found in equation 4,

$$\boldsymbol{z}_{analytical} = \begin{bmatrix} z_{x_1,t_1} & z_{x_1,t_2} & \cdots & z_{x_1,t_T} \\ z_{x_2,t_1} & z_{x_2,t_2} & \cdots & z_{x_2,t_T} \\ \cdots & \cdots & \cdots & \cdots \\ z_{x_S,t_1} & z_{x_S,t_2} & \cdots & z_{x_S,t_T} \end{bmatrix} \tag{7}$$

Next, we take the complex conjugate of $\boldsymbol{z}$ and transpose it, to get $\bar{\boldsymbol{z}}^T$. The complex correlation matrix $\boldsymbol{R}$ is then generated as

$$\boldsymbol{R} = \frac{\boldsymbol{z}\bar{\boldsymbol{z}}^T}{t_T - t_1} \tag{8}$$

where $t_T - t_1$ is the time duration. This matrix has dimensions of $S \times S$ and is complex and Hermitian. This matrix has S eigenvalues λ_s, each with a corresponding eigenvector $\boldsymbol{w}_s$ of length S. Finally, to find the traveling index, a real matrix $\boldsymbol{W}_s$ is generated for each $\boldsymbol{w}_s$, of size $2 \times S$, where the first column consists of the real part of w_s and the second of the imaginary part of $\boldsymbol{w}_s$. Now the traveling index α can finally be found as described above in equation 5.

Amplitude envelope analysis

By analysing the midline motion envelope, the characteristics of different fish can be analysed and compared against each other. To be able to do this effectively, it is important to scale the motions to the body length (BL), as every fish will be of different length. The shapes of the amplitude envelopes can be modelled as quadratic polynomials (Maertens et al., 2017) whose characteristics can be modelled.

(Maertens et al., 2017) employs a travelling wave kinematics including terms for lateral displacement, recoil and steering. This can be presented as

$$\begin{aligned} h(x,t) &= h_0(x,t) + B(x,t) + y_1(x) && (9)\\ &= a_0 A(x) \sin\left(2\pi(x/\lambda - ft + \phi)\right) + B(x,t) + y_1(x) && (10)\\ &= g(x) \sin 2\pi(ft + \psi(x))) + y_1(x) && (11) \end{aligned}$$

where $h_0(x)$ is the lateral displacement, B is the recoil and y_1 is the steering term. The lateral displacement can be described as

$$h(x,t) = H(x)\sin(\omega t - kx) = (a_1 + a_2 x + a_3 x^2)\sin(\omega t - kx) \tag{12}$$

which means it can be described as a quadratic envelope and a sine wave. By studying steadily swimming fish, one can set the steering term to 0, and in this way one can approach the separationg of the recoil from the lateral displacement terms. This was what was attempted in the master's thesis by (Moen, 2020). Such analyses are valuable as they can give input for fish-immitating robotics such as snake-robots and fish-robots.

By ignoring the steering term, one can, such as in (Cui et al., 2018) find the expression for the lateral displacement envelope, $G(x)$, which contains only the pure lateral displacement and recoil terms. The following deduction is closely following that from Cui et al., 2018.

Firstly, to make the motion dimensionless, the wave number is scaled proportionally to the body length,

$$k = \frac{2\pi}{\lambda} = \frac{2\pi}{s_1 L} \tag{13}$$

here, λ is the wave length, and s_1 a constant, which for carangiform and thunniform fish vary around 1.1 to 0.9. To normalize the envelope magnitude at the head of the fish, it can be normalized as

$$\frac{H(0)}{L} = \frac{a_1}{L} = s_2 \tag{14}$$

and, in a similar manner, at the tail,

$$\frac{H(L)}{L} = \frac{a_1 + a_2 L + a_3 L^2}{L} = s_3 \tag{15}$$

thus the constants s_2 and s_3 represent the envelope magnitude at the head and tail, respectively. One last parameter is needed to be able to correctly approximate the envelope, and that is the point in which the lowest amplitudes are reached. For subcarangiform, carangiform and thunniform fish this point is always somewhere in the middle, never at the ends. To find this point, the displacement is simply derivated

$$\left(\frac{\partial H}{\partial x}\right)_{x=s_4 L} = a_2 + 2a_3 s_4 L = 0 \tag{16}$$

Now, the scaled midline motions can be expressed as

$$h(x,t) = \left(s_2 L + \frac{-2s_4(s_3 - s_2)}{1 - 2s_4}\right)\sin\left(s\pi f t - \frac{2\pi}{s_1 L} z\right) \tag{17}$$

By making x dimensionless, $x^* = x/L$ one instead gets a dimensionless version:

$$g(x,t) = \frac{h(x,t)}{L} = G(x)\sin\left(2\pi ft - \frac{2\pi}{s_1}x^{*2}\right) \tag{18}$$

$$G(x) = \left(s_2 + \frac{-2s_4(s_3 - s_2)}{1 - 2s_4}x^* + \frac{(s_3 - s_2}{1 - 2s_4}x^{*2}\right) \tag{19}$$

Where $G(x)$ is the amplitude envelope and the instantaneous lateral displacement is given by $g(x,t)$ as a function of the frequency f and time t, as well as the dimensionless wave number s_1 and position x^{*2}.

1.1.2 Deep Learning

Machine learning is not as new of a study field as some assume, originating in the last part of the last century, but it did not achieve widespread use until the 2000's. The advent of GPU's and better algorithms has resulted in huge improvements in performance. Deep learning (DL) is a subset of machine learning (ML), which again is a subset of artificial intelligence (AI). From Garbade, 2018, we can think of AI as a broader concept of incorporating human intelligence to machines. AI can currently be categorized into general and narrow AI, where general AI can solve problems while narrow AI performs specific tasks.

Machine learning is a subset of AI, and can be thought of as a way of empowering computers with the ability to learn. Deep learning is an efficient way of implementing machine learning, utilizing artificial neural networks. The workings of a deep learning network can be seen in figure 5, where an artificial neural network with one hidden layer can be seen. The name neural network comes from the similarity between this artificial one and the biological version that are found inside our brains. An excelent book introducing to the topic of machine learning can be found in (Géron, 2017), upon which the following is based.

A neural network consists of an input layer, an output layer, and hidden layers. The input layer are simply what inputs the system recieves. In the case of DL networks trained for image recognition, the inputs become the pixels of the image. Although the image is in two dimensions, the input are processed as one-dimensional. This is achieved by simply lining every row of pixels after each other.

The next layer, after the input layer, is the hidden layer. This layer is not restricted by the size of the input layer, and can be larger. There can be as many hidden layers as the user want, some of the leading networks today typically have everything from very few to between one and two hunder. Such examples are ResNet34 and ResNet152 (He et al., n.d.). The total activation value of each layer amounts to 1, or 100%, as indicated in figure 5. Note that there are two inputs, one hidden layer with 4 neurons, and three outputs. The last layer is the output layer, which is the classes that the network is tasked to detect. In the instance of object detection, this could be humans, cars and stoplights, such as is already in use by Tesla electric vehicles.

The way that the neural pathways are trained is by giving the network already annotated data. In this way, both ends of the network are already determined, and the pathways between them can

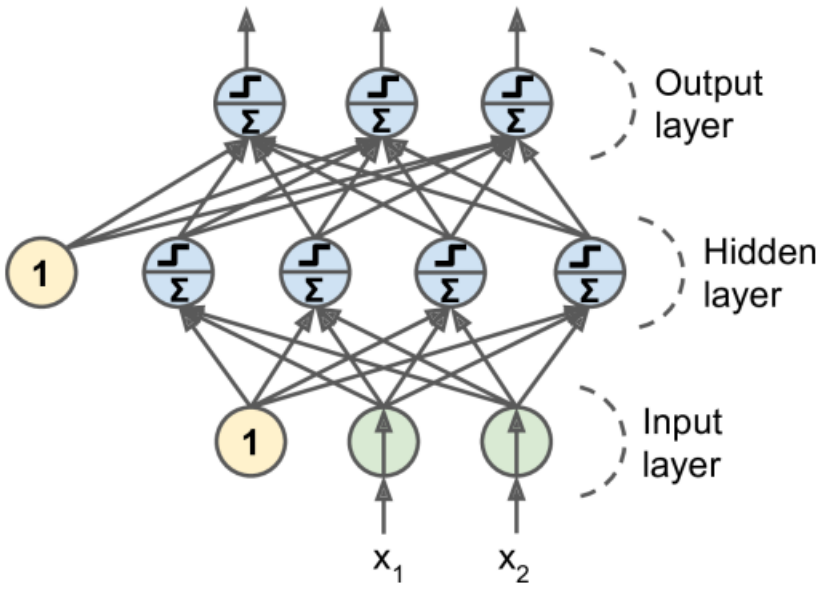


Figure 5: An example of an artificial neural network. Also known as a deep neural network, when the number of hidden layers increase 1. Figure from Géron, 2017

be resolved to give the correct answer. When this is done enough times, each individual neuron will be given a preferrence to the ones that it connects to. This weighting is what it means when a network is said to be trained. This can compared to how our own brains learn, where many tries has to be done before an understanding of a certain subject will form.

1.2 Preliminary works

1.2.1 Deprecated method: mask generation using openCV

There are many possible ways to implement a way to extract fish locomotion from video. In the project thesis, the first method examined was using OpenCV without any machine learning. As a starting point, a background subtraction scheme using OpenCV in python was implemented. Further, a Canny edge detector was also applied. Both of these yielded insufficient results.

1.2.2 OpenCV Results

By inspecting the results presented in figure 6 it is clear that neither the Background Subtraction schemes nor the Canny edge detection schemes are good enough for the purposes in this report. For the background subtraction, the MOG2 scheme was applied, experimenting with different parameter values. The history was varied from 5 to 500 and the variance threshold was varied from 5 to 30. All resulted in variations of the image presented in figure 6, in which a history of 5 and a variance threshold of 10 were used. For lower variance thresholds and histories, more of the image is deemed to be foreground. Similar results were found using the MOG and KNN schemes as for MOG2.

There are mainly three problems in using the OpenCV background subtractions as shown. Firstly,

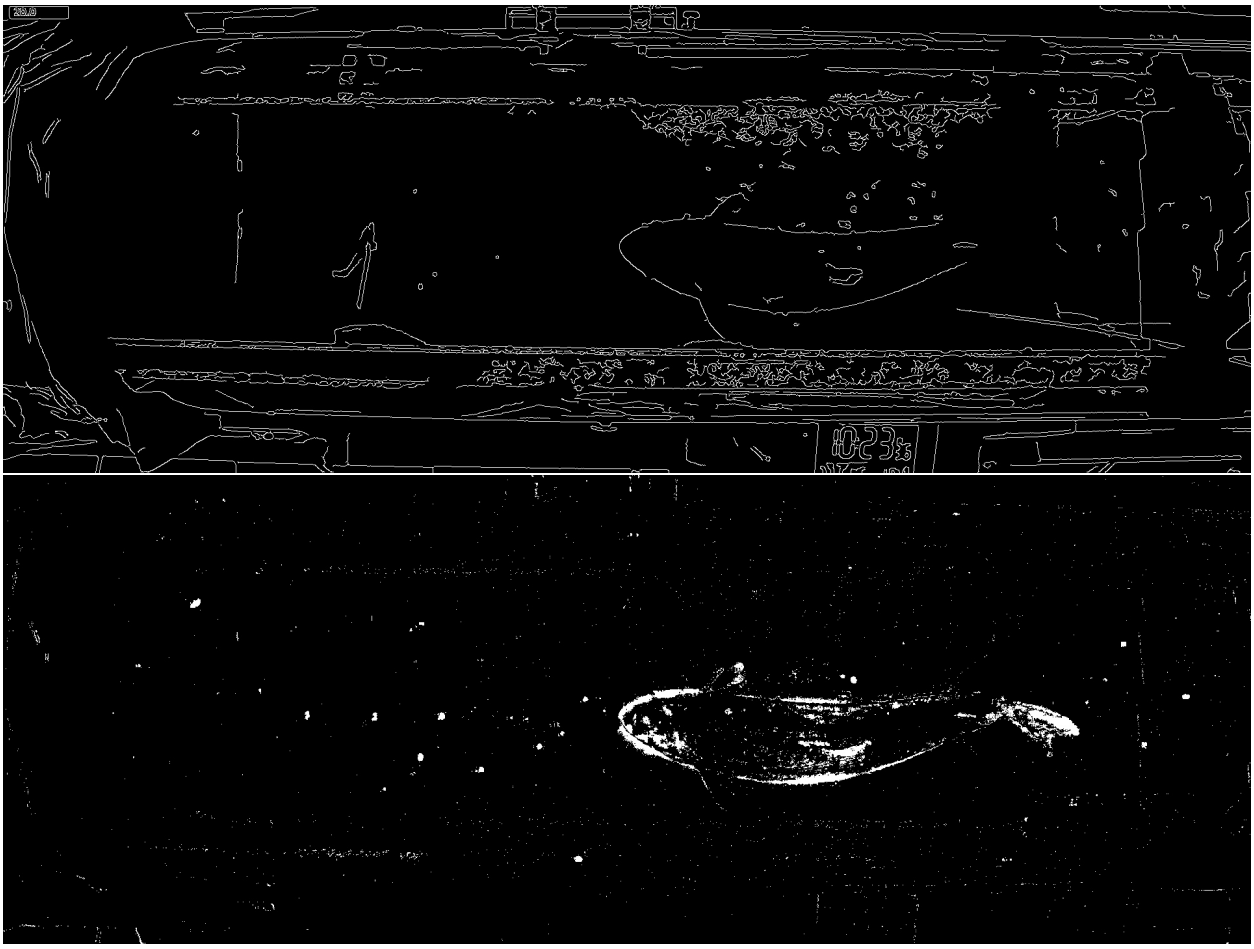


Figure 6: Top: Canny edge detection algorithm. Bottom: MOG2 background subtraction. Notice that the fish is not fully defined in either method and the background is still featured considerably. Implementing a midline extraction on either of these images would be difficult.

the background is almost never completely subtracted, mostly due to shaking of the camera and the rippling effect of the flowing water. Secondly, larger bubbles are counted as foreground, which they are, even though it is not relevant for us. Thirdly, the fish does not oscillate in a uniform speed, and so some parts of the fish seldom move and thus is more easily recognized as background, while other parts move quickly and are more easily recognized as foreground. As a result of all these issues, the end result is a predicted foreground predicting of too much of the background, bubbles which are not of interest, and too little of the fish.

The Canny edge detection algorithm is quite good at picking up the fish. It is suspected that even more of the fish, or possible all of it, could be recognized if the parameters were tweaked more. However, the main issue here is that the edge detection does not differentiate between fore and background. Therefore all edges in the video are found, including a lot of the background. In addition, the rippling effect of the flowing water is also detected as edges, which gives a lot of noise in the picture.

1.2.3 Method Conclusion

In conclusion it can be said that both the background subtraction and the edge detection algorithms are promising, but to make them work it might be necessary to combine them, or to use them in conjunction with some bounding box or object detection method as well. This is what led to the methods implemented in this thesis.

2 Material

In this thesis, only one set of video recordings of swimming fish experiments were used. The experiment was conducted at NTNU/SINTEF Sea Lab in Trondheim, Norway in 2019. They were conducted in a Blazka-type swim tunnel of approximately 59L. The dimensions for the area the fish was able to freely move in was 120×16.5 cm. The aim of the experiments were ”verification of model for swimming movement of salmon" (Xu, 2019).

By inspecting figure 7, the experiment setup can be viewed. The setup consists of a cylindrical tank, with an inlet (to the right) and an outlet with a grid covering it, to prevent the fish from being pushed downstream (to the left). In the captured video, the directions are reversed, so the inlet is to the left of the image. The camera placement is directly on top of the fish, but the distance is not known. It is assumed that the physical dimensions of the objects captured on video are captured without distortion. It should be noted that the tank containing the fish is cylindrical, so some optical distortion will occur as the fish swims close to the sides. In addition, there are two plastic walls inside the swim tunnel, preventing the fish from swimming all the way to the sides of the tunnel. The walls and the cylindrical form of the tube results in some distortion at the sides of the tunnel.

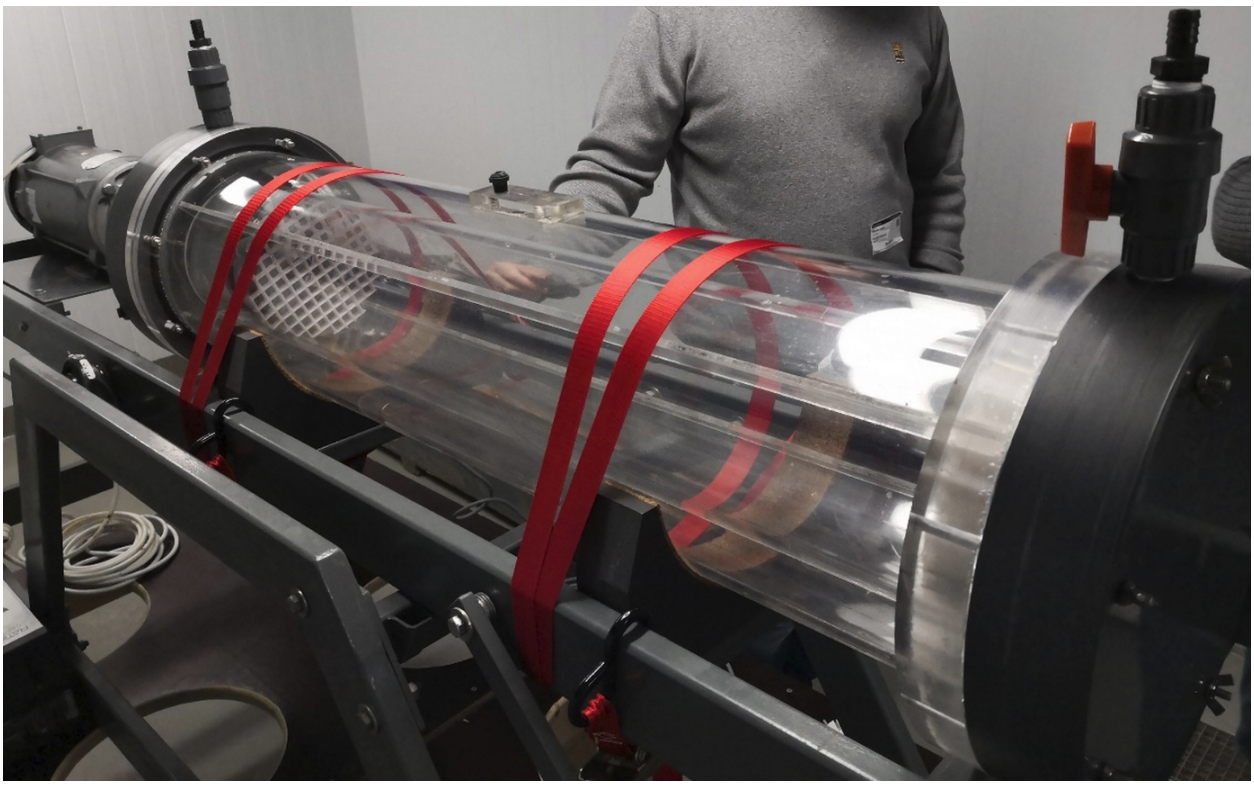


Figure 7: Image of the experiment setup.

To minimize stress, the salmon were places in the experimental setup for 6-7 hours before the experiment started so they could acclimatize and calm down to the new environment. It is believed that stress is a large factor in such experiments and could affect the natural behaviour of the fish. As the fish used were laboratory grown, they were not as capable swimmers as wild salmon, and

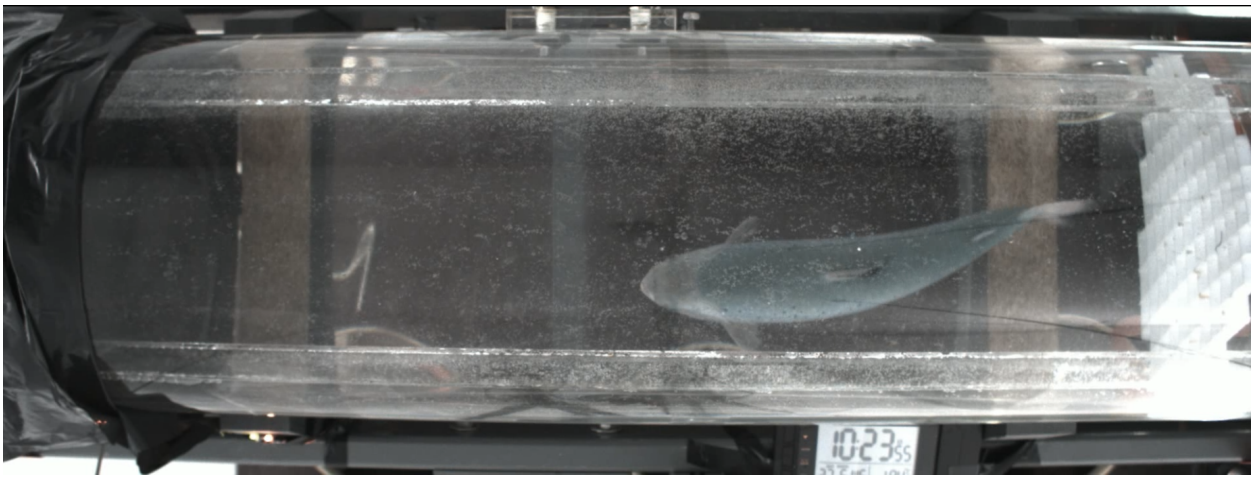


Figure 8: Screenshot of the recorded video in the experimental setup

Table 1: The flow velocity for the specified times

Fish no.	0.5BL	1BL	1.5 BL	2 BL	2.5BL
5		10:15	10:27	10:35	10:56

could not swim faster than around 2 BL/S, meanwhile wil salmon are capable of reaching almost 3BL/S. So as not to startle the fish, and to induce as natural behaviour as possible, the inflow velocity was only increased by increments of around 0.5BL/S, and only after the fish had shown steady swimming for 5 minutes.

The camera capturing the video is set up directly above the fish so that it can capture the entire space that the fish can move in, without panning. By observing figure 8, a screenshot of the video can be seen. Information of the video material can be viewed in table 2. It should be noted that frame rate was not completely consistent, but varied around 48 ms. Likewise, the physical dimensions of the fish used in the experiment can be viewed in table 3. Lastly, the flow speed can be seen in table 1. Note that only fish number 5 is mentioned, this is because this is the fish that was selected for use in this thesis, as the other fish did not provide as clear swimming behaviour as fish number 5.

Table 2: Data on the video from the experiment. Time is given in milliseconds

Resolution	Recording time	Recorded Frames	Resulting Framerate	Time between frames
1920×720	3e+06	149990	48.41	20.65

Table 3: Descriptions of the fish used in the experiments. Length and height are given in cm and weight in g.

Fish number	Fork length	Height	Side	Weight
1	43.5	11.5	6.5	1153.6
2	43	10.5	6	1127.5
3	37			
4	42.1	9	4	932
5	29	6.2	2.5	319.5

3 Results

3.1 Mask identification

To be able to convert videos of fish swimming into usable datasets, a machine learning deep learning network was trained. The main advantage to using deep learning networks is that once trained they make acquiring new datasets very simple, the end result in this thesis was that videos could be processed directly, the only requirement being that the video snippets be of suitable length and quality.

The deep learning network chosen in this thesis is Detectron2 (Yuxin Wu et al., 2019), described as 'Detectron2 is Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms.' on the official detectron2 github repository (Yuxin Wu et al., 2019).

There are two sides to using machine learning, one is training a suitable model on manually annotated data so that it can learn to recognize what you want it to, and the other is using the model on raw, unannotated data. In this thesis, both steps were done. To get an understanding of the process of the systematic analysis method, refer to figure 9.

It is important to stress that although deep learning has been successfully implemented in this thesis, the process of actually training a deep learning model is still treated as a 'black box'. Machine learning is a different field of expertise than hydrodynamics, so it is used as a tool. Therefore, the verification of the results of detectron2 are mainly visual, and in the 'Verification' section the end result is analysed to verify whether the results are usable or not.

3.1.1 Installation

Detectron2 is recommended to be run in a Linux or macOS enivornment. In addition to this, for calculations to be performed at optimal speed, they need to be run on the graphical processing unit (GPU). In detectron2's case, it is optimized for utilizing CUDA, which is a parallel computing platform, for use on NVIDIA GPU's. The choice of linux operating system naturally fell on Ubuntu as it has good support for CUDA. In addition to this, detectron2 needs Python as well as PyTorch, torchvision and OpenCV. The official installation page[1] should be consulted for an in-depth description of the installation process. Installation will not be trivial if you do not have experience with Linux, python or the command line.

3.1.2 Training a deep learning model

To start, a model had to be trained, and as such it needs annotated images to train on. Luckily, it is possible to add on to previously trained models, which can be accessed within detectron2's own api. It should be noted that details pertaining to the different models one can use is hard to obtain, so it can be difficult to know exactly what data a specific model is built on. For reference, the model used as a building block is accessed as 'COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml', and is known as 'model_final_f10217'. The COCO dataset is a

[1] https://detectron2.readthedocs.io/en/latest/tutorials/install.html

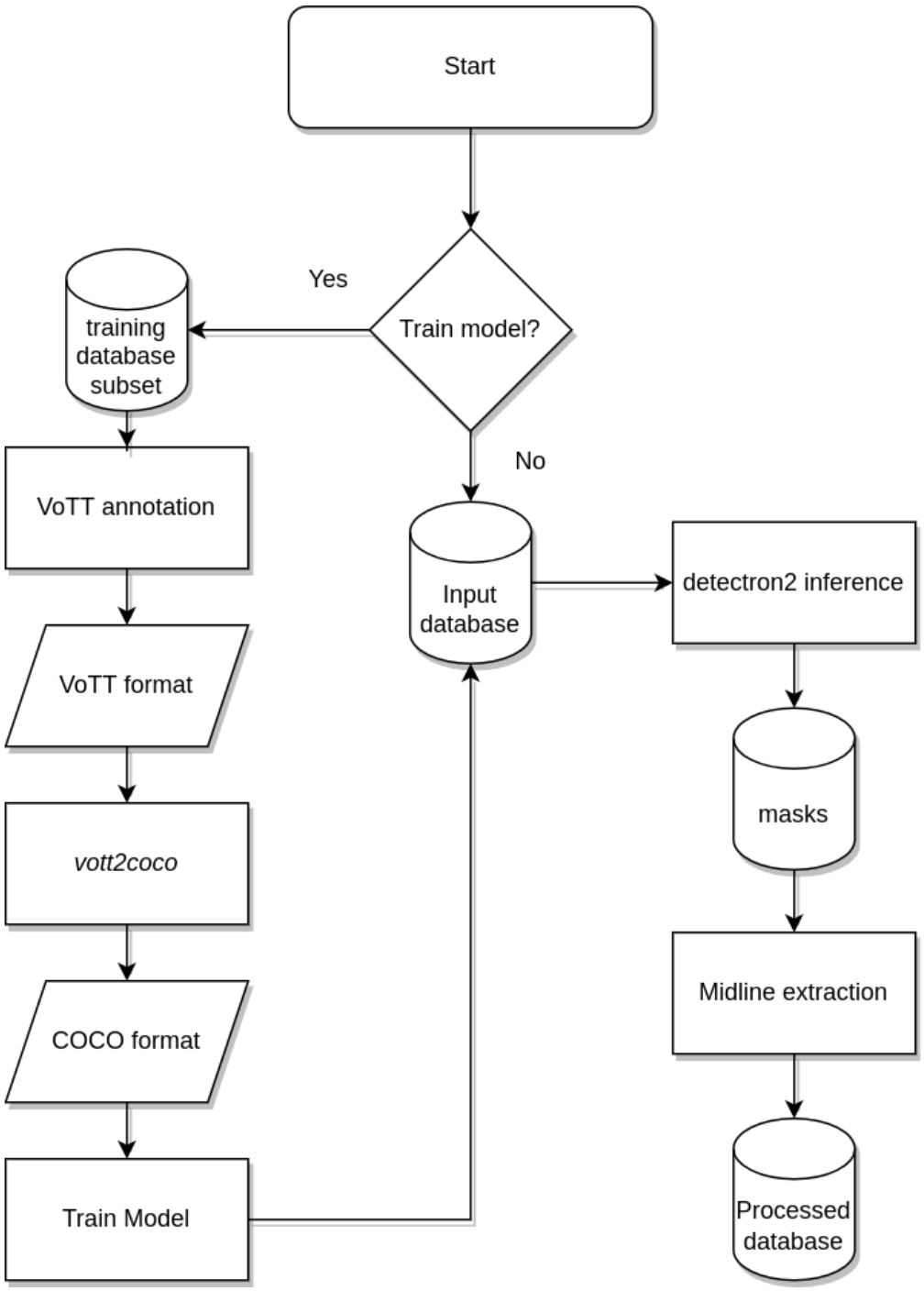


Figure 9: Flowchart detailing the process for systematic extraction of locomotion used in this thesis.

large and well respected dataset used as a frame of reference for many of the leading machine learning networks.

The model was chosen as it is provided in the detectron2 documentation, (Detectron2, n.d.) and instantly provided good results. Futher training of the is done by adding a manually annotated dataset of the fish from the video later that is to be analysed. Models trained on the COCO dataset have already been trained with huge amounts of data.

Preprocessing data

To be able to run a deep learning on the fish experiments, it is necessary to do some augmenting to the data. To train the neural network it first needs manmade input in the form of annotations and segmentations of fish outline. In the figure 9, detailing the pipeline of workflows, the steps 2 through 5 detail this work.

VoTT: Video annotation software

There are many ways of annotating images or videos, the chosen method in this master's thesis is by utilizing Microsoft's Visual Object Tagging Tool, which is "an open source annotation and labeling tool for image and video assets" (Microsoft, 2019), which includes the ability to export annotated datasets to local storage.

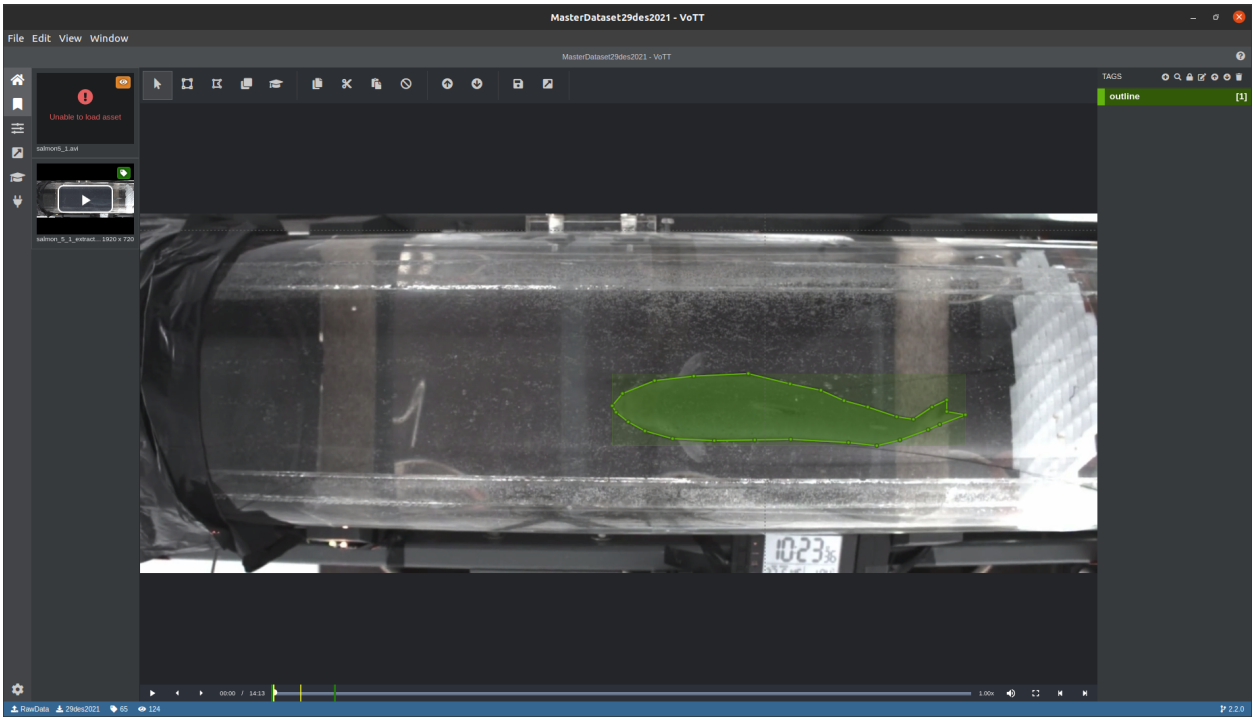


Figure 10: Screenshot of an segmented annotation of a fish used for training detectron2. Note that 27 points have been placed manually to outline the fish, to give more accurate training data for the neural network training process

By inspecting figure 10 an one can see how the annotation process is carried out. Note that the

pectoral fins of the fish are not part of the annotation and are ignored henceforth, as they are not necessary for approximating the midline of the fish.

Converting dataset format to COCO format

COCO, which stands for ´´Common Objects in Context", is a well know and highly regarded dataset containing 330 000 images, with 200 000 of them being labeled. It is a ´´Large-scale object detection, segmentation and captioning dataset"(Lin et al., 2015)

For detectron2 to be able to recognize and train on the annotated images, they can either be ported to a format that detectron2 already recognizes, or alternatively, new methods can be written for detectron2 to recognize whatever format they already are in. Luckily, there exists such a reposirory, found on github, called VoTT2COCO (Ptak, n.d.). By utilizing this option, detectron2 must be set up to take a custom dataset in the COCO format, which makes training easier.

To use the script, the setupfile must be edited so that the source "path" variable goes to the dataset in VoTT-format, which must contain one directory labelled "images" with all the images and one directory with all the annotations, labelled "annotations". The destination "path" variable must point to a directory containing one directory labelled "images". The script will generate annotations.json in the destination directory, a single file containing all annotations.

Training

This is a thesis in hydrodynamics, and some parts of the machine learning implementation process must be regarded as a ´black box'. Some parts of the setup of the detectron2 model config is therefore treated as a ´black box', and as the end result is good enough for the scope of this thesis, are not investigated further. Note, however, that if the methods of this thesis are to be further examined, this section in particular should be examined further, as it is probable that changes to the configuration of the setup and implementation of the training can result in better segmentation masks, and thus better midline approximations.

By setting up the config of the model to be used, it is possible to adjust certain parameters to adjust how it is to be applied. The ´´num_workers" option is used for setting the number of CPU cores to use. In this case, as the CPU has 16 cores, half of them are allocated to the deep learning task. It is important to not use all of them, as this will make the computer crash. A key aspect of choosing the current setup of Linux and detectron2 is to use the CUDA capabilities it offers. The time difference for training to be performed is big, although luckily building upon previously built models lessens the need for training considerably. In this case, a model was trained on a dataset consisting of 28 images on the CPU, which took around 28 minutes. In comparison, training a model on the GPU took around 1 minute and 20 seconds. This results in an increase in efficiency of around 20 times! The number of images per batch, max iterations, and batch size all impact training time and also depend on the amount of RAM the GPU has.

To choose a model to build upon, the line ´´cfg.merge_from_file" calls on a model. There are several models to choose from which are incorporated into detectron2's structure. It is also possible to call on your own models. Note that the chosen model is based on the COCO format, which is the same as the format that the annotated data was converted to. Specific information on the

different models are hard to find, but the COCO datasets are well regarded and used throughout the machine learning, as detailed aboce in section 3.1.2.

```
    cfg = get_cfg()
    cfg.MODEL.DEVICE = "cuda"
    cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/
                                                  mask_rcnn_R_50_FPN_3x.yaml"))
    cfg.DATASETS.TRAIN = ("salmon_train",)
    cfg.DATASETS.TEST = ()
    cfg.DATALOADER.NUM_WORKERS = 8
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(
        "COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let
                                                    training initialize from model
                                                    zoo
    cfg.SOLVER.IMS_PER_BATCH = 6
    cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
    cfg.SOLVER.MAX_ITER = 400
    cfg.SOLVER.STEPS = []  # do not decay learning rate
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
    os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
```

3.1.3 Inference

Inference is when a model generated by training a neural network is applied to an image, and it then guesses what is in the image. In our case, the model used is built upon already made models, where a new class of fish outlines has been created. This means that the model will be able to identify these types of fish outlines as well as every other class it has been trained on.

To use add the custom model trained previously, it is necessary to include the config setup as described in the previous section, as it sets up detectron2 to work in the desired way. In addition to this, the previously trained model must be loaded.

```
cfg.MODEL.WEIGHTS = os.path.join("output/Models/30des2021model.pth")
```

Once a model has been loaded, using it for inference is a matter of a few lines of code.

```
# path to the model just trained
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
predictor = DefaultPredictor(cfg)
from detectron2.utils.visualizer import ColorMode
dataset_dicts = get_balloon_dicts("balloon/val")
for d in random.sample(dataset_dicts, 3):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=balloon_metadata,
                   scale=0.5,
                   instance_mode=ColorMode.IMAGE_BW
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(out.get_image()[:, :, ::-1])
```

As can be seen, using inference is simply a matter of writing a few lines of code. The real challenge lies in understanding the entire framework and setting everything up correctly.

3.1.4 Mask results

The pure output produced by detectron2 needs some work before it can be imported into the midline extraction algorithm. First, let us investigate how detectron2 performs on the chosen dataset. The object detection and segmentation algorithm works very well, except for the two problem areas of the tail and the head. To be able to accurately capture the locomotion of the fish, both these areas need to be captured. Luckily they are captured every time, it is just that some information is lost which will affect the end result.

By looking at figure 11 the masks that detectron2 generate can be viewed. The images are chosen as they represent both the strengths and the weaknesses of the trained model. It is clear that the nose outermost part of the nose of the fish is not accounted for, however the discrepancy here is mostly small, there are only some instances where this is significant. The greater discrepancy is in the tail region. In the first image the tail is fully represented, in the second one the trend is represented but the mask does not cover all. In the third case, the fin is bent in such a way that it is pointing opposite of the body part of the tail. This is not represented, and will probably affect any analysis that is performed, however to what degree is hard to say.

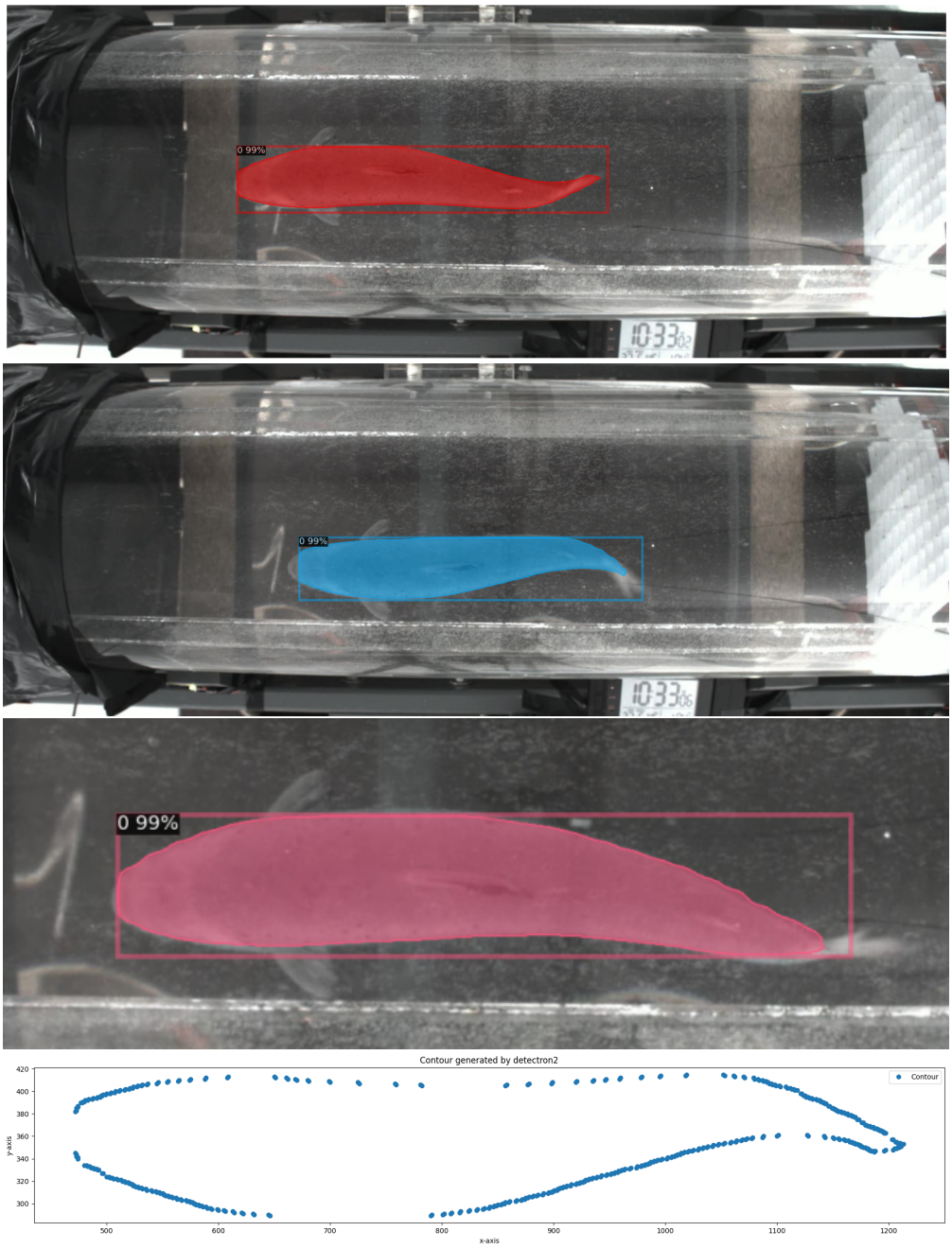


Figure 11: The top three images show how detectron2 applies masks, on top of the image the mask was generated from. The bottom image shows only the points making up the mask. Note that the bottom image does not correspond to the above images.

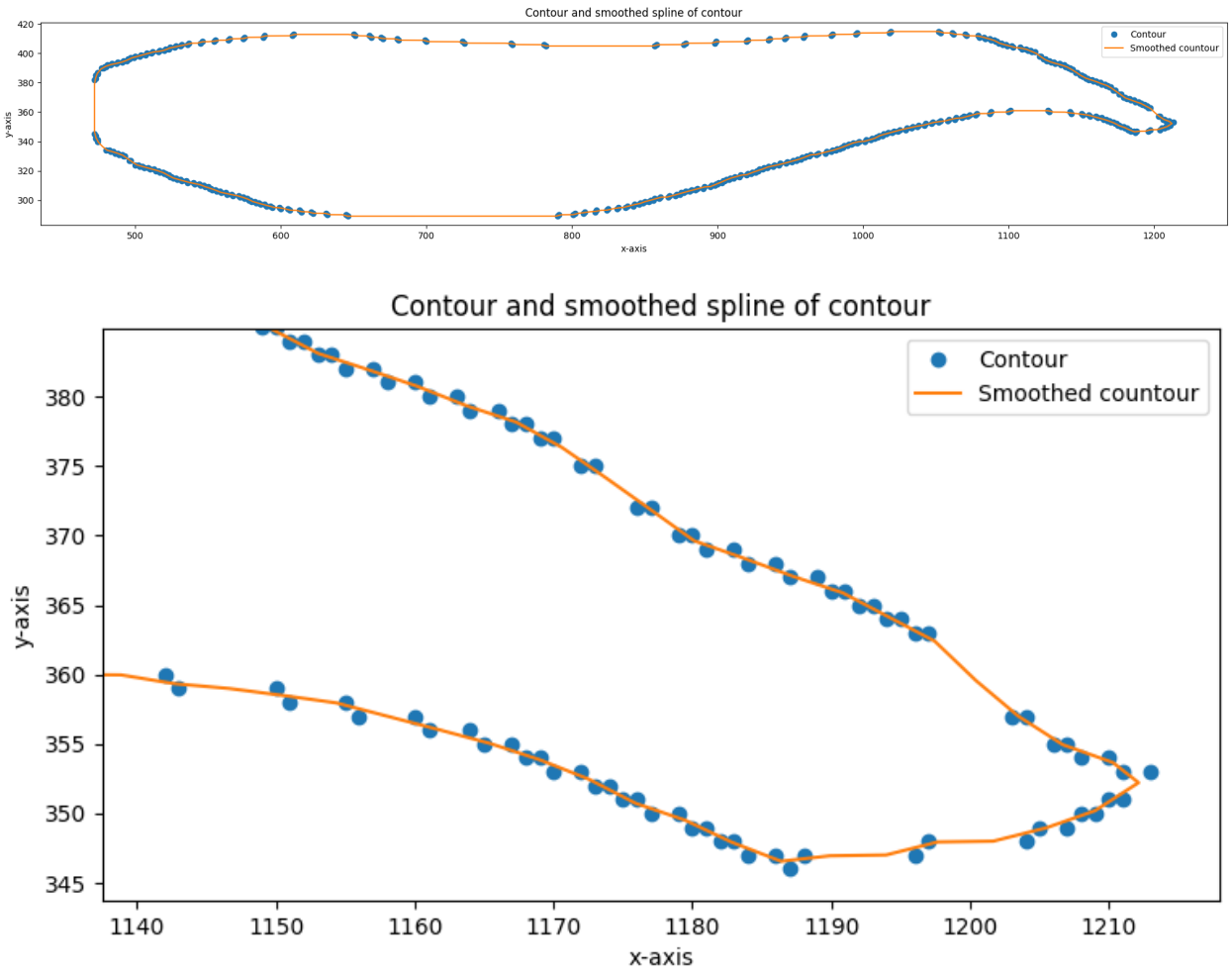


Figure 12: The data points of the mask applied by detectron2

Mask post-processing

By inspecting the last figure in 11, it is clear that the masks are sometimes lacking points in certain areas, also areas that are well represented when the masks are overlaid on their corresponding images. This would become a problem later, when the midline is to be extracted, as an even distribution of points is necessary. To combat this, a smoothing spline is made, and new points are made along this spline. To create such a spline, it is necessary that the vector containing the mask points is made to be closed. As such, the first point of the python array containing the masks are appended to the last.

The spline is then made using the scipy library of python, utilising the scipy UnivariateSpline function. From the scipy docs, the spline is a ”1-D smoothing spline fit to a given set of data points´´ (scipy, n.d.). The spline can be of order between 1 and 5, with the default being a cubic spline (k=3). The best results were found by setting k=1. The smoothing factor s was not found to have much impact on the smoothness of the spline, this might be because the order was linear and the distance between each point low. The spline was applied as following:

```
splines = [UnivariateSpline(distance,coords, k=1,s=s) for coords in points.T]

alpha = np.linspace(0, 1, N)
points_fitted = np.vstack(spl(alpha) for spl in splines).T
```

where distance is the calculated distance along the coordinates, N is the number of new points, set to 400, and points_fitted are the smoothed coordinate points. The new, smoothed masks are vastly superior to the raw output of detectron2. By inspecting figure 12 the difference becomes apparent. Not only are all points now evenly spaced around the entire circumference of the mask, the points are also smoother, as can be seen in the lower of the two images. In addition, the large, empty areas have been filled in by straight lines.

3.2 Midline extraction

3.2.1 Medial Axis Shrinking Ball Implementation

Motivation

By utilising a method capable of finding the midline on a fish moving in any direction, analysis of much more complex movements than simply a fish moving in a fixed direction are possible. As such, the methods in this thesis could be built upon to analyse fish moving in more complex manners.

Installation

The algorithm implemented was based on the method described by (Ma et al., 2011), titled ”3D Medial axis point approximation using nearest neighbors and the normal field´´. Researchers at Delft University applied in 2014 the this method in a python repository found on GitHub, titled masbpy (Peters & Ledoux, 2014). In 2015 a faster, more robust version was made in C++ instead, also found on GitHub (Peters et al., 2015), called masbcpp. Note that while both of these repositories were made for 3D implementation, they can also work for 2 dimensions.

Firstly, the C++ implementation was attempted to download and run. Immediately there were problems, as it was not possible to install the software. After some cumbersome googling, a solution was found and the program was ran on a simple tutorial case. The program did not run and did not give any useful information out to the user, so debugging was very difficult to do. It was therefore decided to instead try the python version.

The masbpy repository also presented immediate problems. These were possible to debug, however, so it was decided to try. The masbpy repository is written in the not supported python2, which has been deprecated in favour of python3. Luckily, there exists certain scripts to convert python2 to python3, this is included in the tools/scripts directory of python3 and is called by

```
$ 2to3 --output-dir=python3-version/mycode ...
                            -W -n python2-version/mycode
```

After this had been done, many more errors were encountered. After a time-consuming effort had been conducted, a working version of the program was made.

Method

Masbpy works by taking in a set of input coordinates, and outputting the set of internal and external midpoints corresponding to them. To do this, it first generates a set of normal vectors for each point, and then based on this a ball is made whose tangent is at the relevant point. This ball is then shrunk until it only touches one other point in the structure. The process is illustrated in figure 13, where the normal vectors are overlaid on the coordinate points making up the structure. Then, shrinking balls are laid for each point, one pointing internally and one externally. In this way the internal and external points are found.

In the picture to the left of figure 13, the internal points are found, the algorithm has started at the top, worked its way down the right and is now moving towards the underside towards the

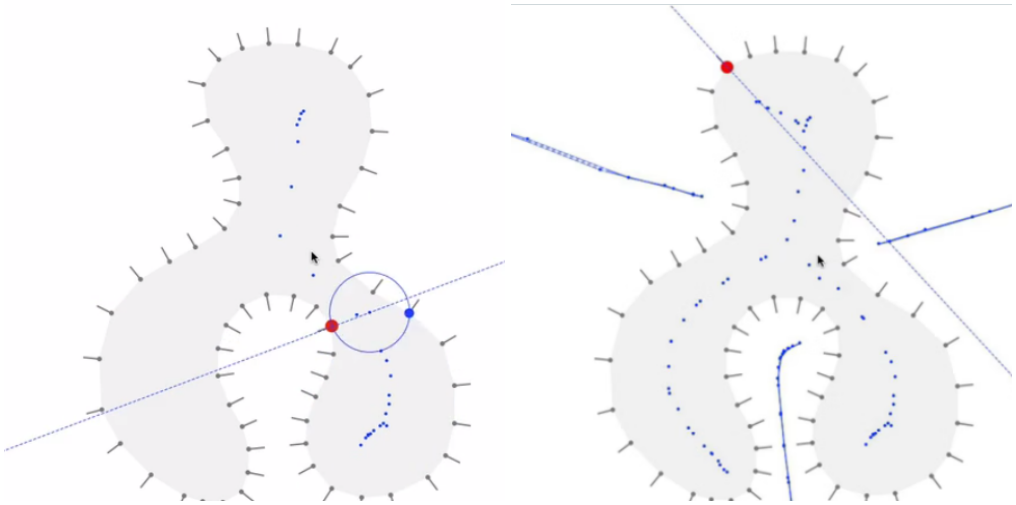


Figure 13: Example of how the masbpy algorithm works. These two images are screenshots of the example video by Peters (Peters, 2014)

left. In the picture to the right, almost all internal and external points have been found, and the complete results can be seen. The lines through the working coordinate (red point) indicate the direction that the normal vector is pointing. One side is the internal and the other the external direction.

Implementation

The first thing that must be done is to make the normal vectors. This is done using k-d trees, which is a way of searching for nearest neighbors in both 2- and 3-dimensional space. In the application of the masbpy application, this was done by the compute_normals function:

```
def compute_normals_my_func(coord, k=10):
    kd_tree = KDTree(coord)
    neighbours = kd_tree.query(coord, k + 1)[1]
    neighbours = coord[neighbours]

    p = Pool()
    normals = p.map(compute_normal, neighbours)
    normals = np.array(normals, dtype=np.float32)
    return normals
```

The main problem encountered was that the normals would often not point in the correct direction. This is due to how the k-d tree works. Depending on how many neighbours were chosen, different results were achieved. Sometimes, the normal direction alternated 180 degrees every other point, but the more reliable results were of a nature akin to that presented in figure 15. There, most of the normals are correct, but in the nose and tail region there are errors. If the only issue was the vectors pointing in the directly opposite way, i.e. the internal and external directions being switched, finding the midline could still be achieved by choosing the midline as only the points inside the body boundary. In the head region the normal vectors are pointing almost tangentially to the outline, producing garbage results in those cases.

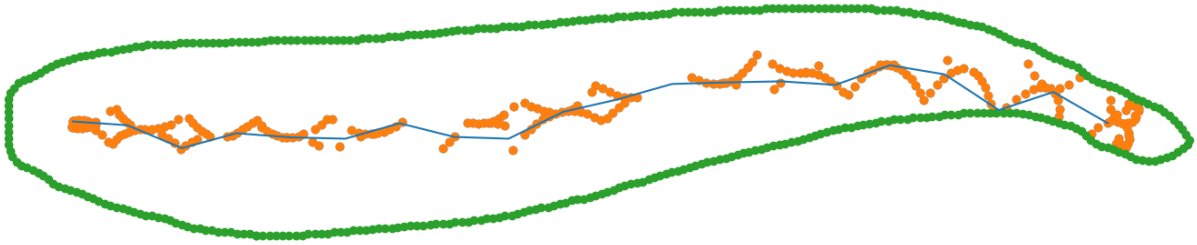


Figure 14: Initial result of using the masbpy algorithm. Too many coordinate points results in inaccurate midline approximation.

After the normals had been computed, another problem was encountered. The shrinking balls did not operate correctly. As there were too many points, they shrunk so much that they did not reach the opposite side of the fish, but instead got so small they touched their actual nearest neighbors. This resulted not in midline points but simply a set of points near the mask coordinates. This effect can be seen in figure 14, where the midline points do not line the outline coordinates perfectly as described, but rather form patterns based on imperfections in the outline points and which happened to be the closest. Such a result is not usable in any further analysis.

As the results by the methods described above were non-satisfactory, one last attempt was made. As the shrinking balls were touching the neighbors, a solution was sought by decimating the coordinate vectors. In this way, the set of 400 points were reduced to 15 points instead. This gave results such as in figure 15.

The approach comprised of subdividing the coordinate points of the fish outline into N arrays, each array containing every N point in the original prediction mask from detectron2. For each array a midline was calculated, which were then superimposed together to form one midline for each frame. The idea is that the fewer number of coordinates in each subdivision will yield clearer results. To counter the loss of information by creating vectors of only every Nth element, the combined midline points were fitted by linear interpolation. To put it simply, for the original coordinate vector containing N points, N/n different variations of figure 15 were created, using n different coordinate points each. Then the midline points from all these N/n variations were combined into one midline.

Steps detailing the final method of implementation

1. import coordinates
2. create normals vector from original coordinates
3. decimate normals vector into N/n vectors containing n points
4. produce midline points for each decimated normal vector
5. combine all midline points
6. filter midlines

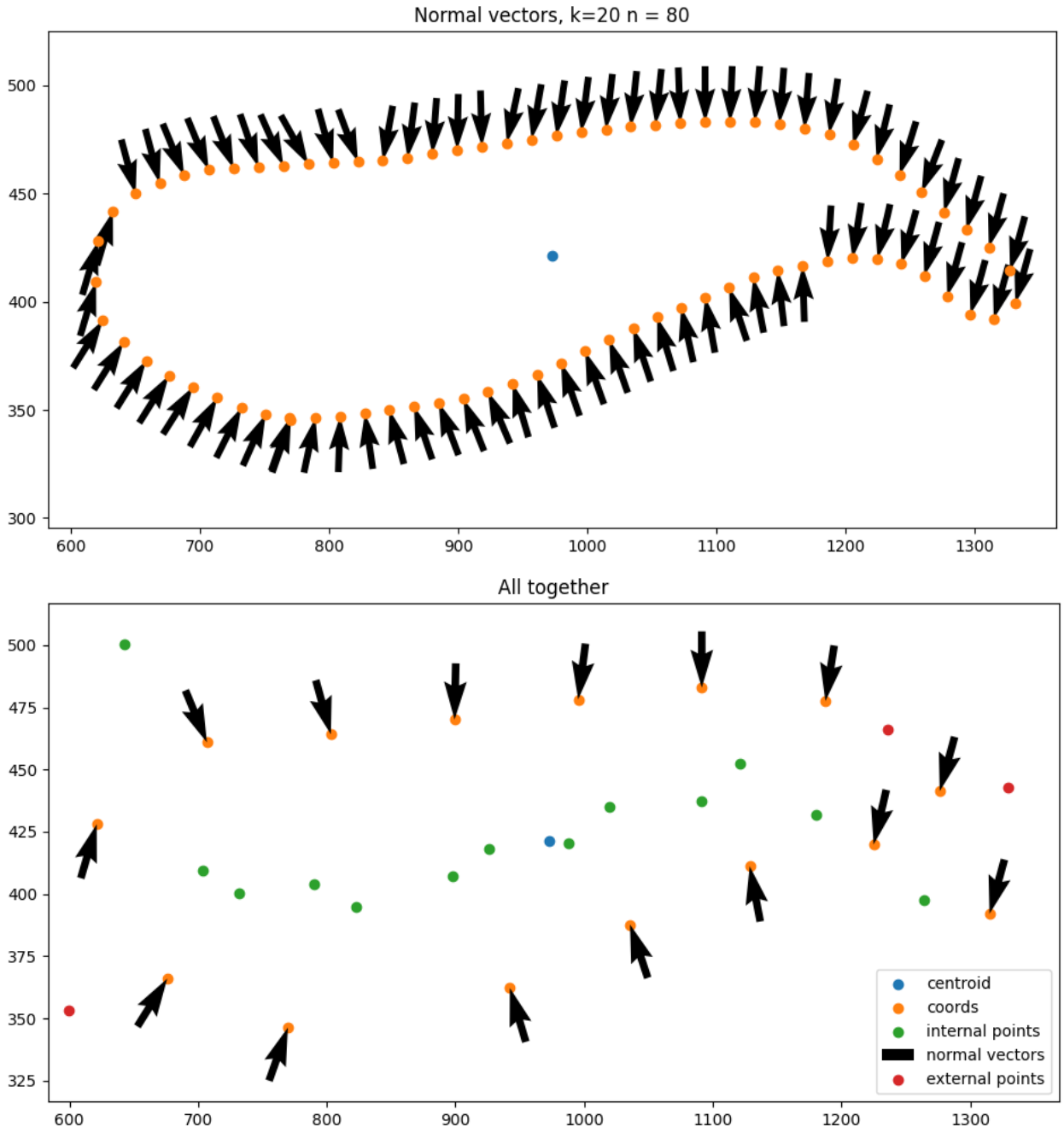


Figure 15: Masbpy results for a decimated coordinate array. Note the algorithm is able to predict a midline in the center of the fish

Final results

By inspecting figure 16 it is clear that the masbpy method results in uneven result even when substantial effort has been made to correct for the weaknesses of the method. The end results are better than what was initially achieved, and in figure 16 the area around the dorsal fin is predicted quite well, however the rest of the midline exhibits too much variation for a stable result. It was therefore decided to drop this method in favour of an easier one.

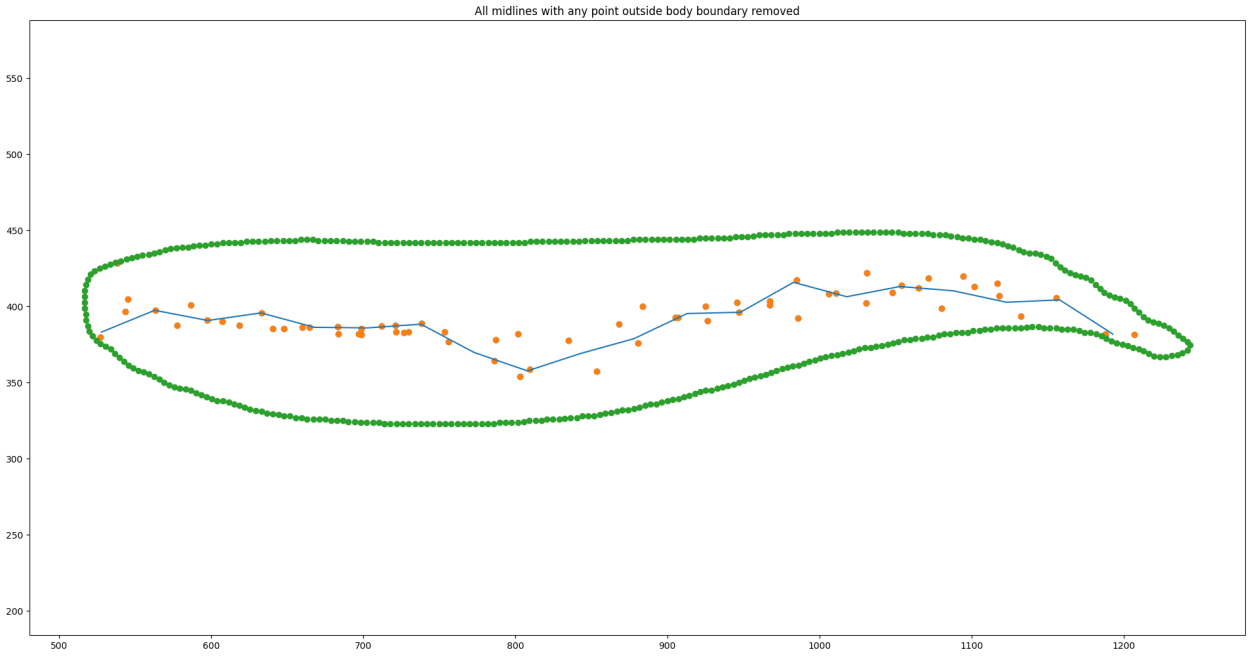


Figure 16: Implementation of computing normal vectors for the fish outline produced by detectron2. Notice how there are errors in the computations in the tail and head region.

3.2.2 Simple approximation

As more complicated methods were not working, a simpler way to estimate the midline was implemented. By dividing the fish into n sections, it is possible to simply find the average x- and y-position for each section. This means that the resulting midline will contain n-1 points. The method is extremely fast and fairly accurate. The midline rib approximation method requires the fish to be aligned alond the x-axis, thus it cannot swim in arbitrary directions.

Method description:

Below is printed the method for extracting the midline from each individual mask, calculated for each image frame by detectron2.

```
1. set start values
2. import coordinates ported from detectron2
3. calculate centroid based on coords
4. calculate midline based on coords
    4.1 Create x-vector of N+1 points over the length of the fish
    4.2 find each point in the coordinates where the x-coordinate
        is within the current interval of the x-vector generated.
    4.3 set midline point as mean of x- and y-values in each of
        the N intervals
    4.3 if interval is at the start or end of the fish, set the
        x-value equal to the end-point, and the y-value equal to
        the mean
5. calculate angles and change of angles between each midline
   point. (Unused feature)
```

Another limitation that must be addressed is how the endpoints are calculated. Initially they were simply calculated as the mean x- and y-position, this resulted in the head and tail not being properly approximated. The problems with this is that the fish is a fixed length and so the midline must not change in length, in addition the tail especially is hard to accurately predict and the last part may not be accurately accounted for in this method. The method employed instead, see fig. 18, aims to account for constant fish length but as a result gives skewed endpoint directions. Due to time concerns better alternatives could not be produced.

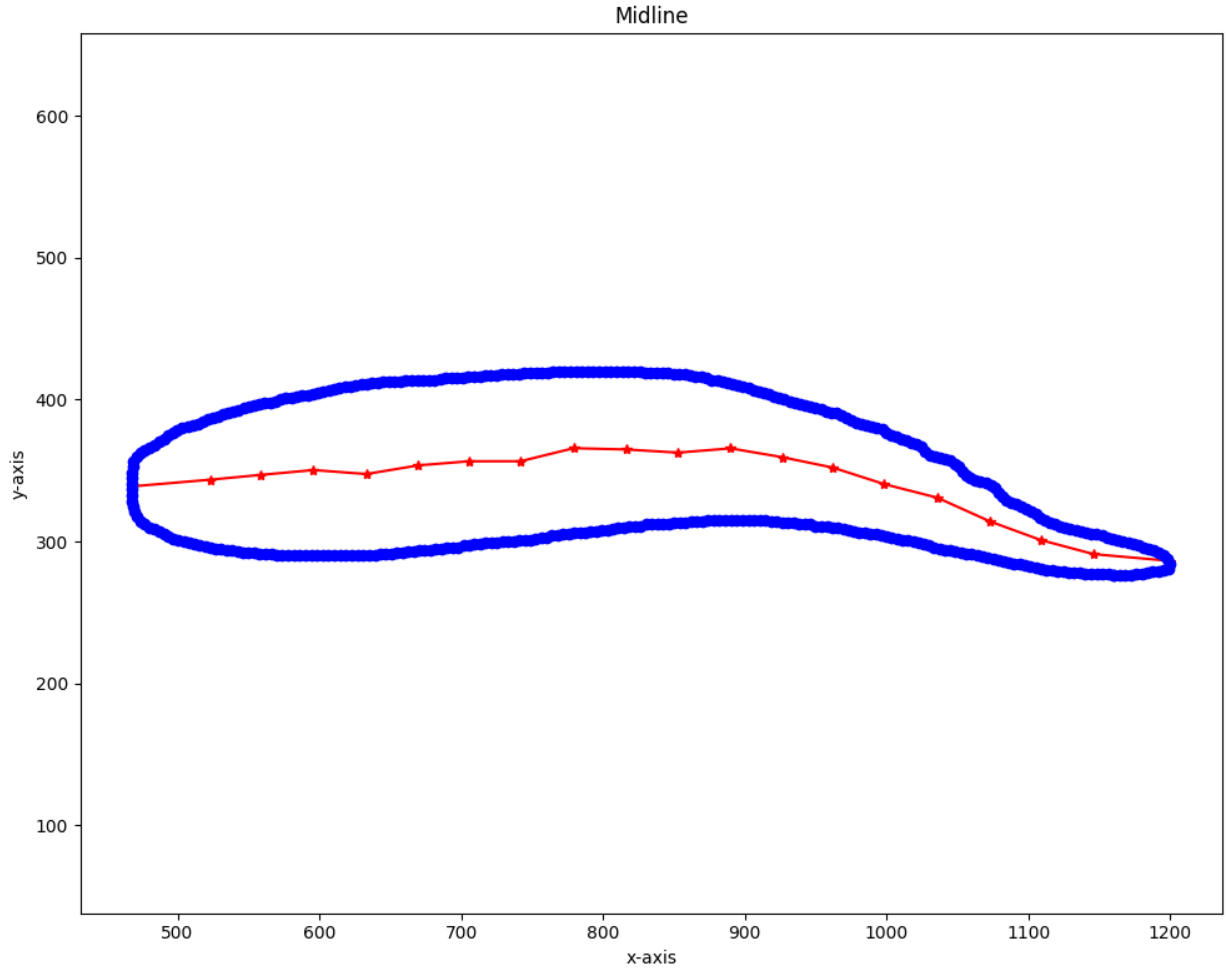


Figure 17: The fish body is divided in n points, and for each section, the mean point is found as in the first, leftmost section. The result is shown as x's. Note how this method does not incorporate the extremes of the fish.

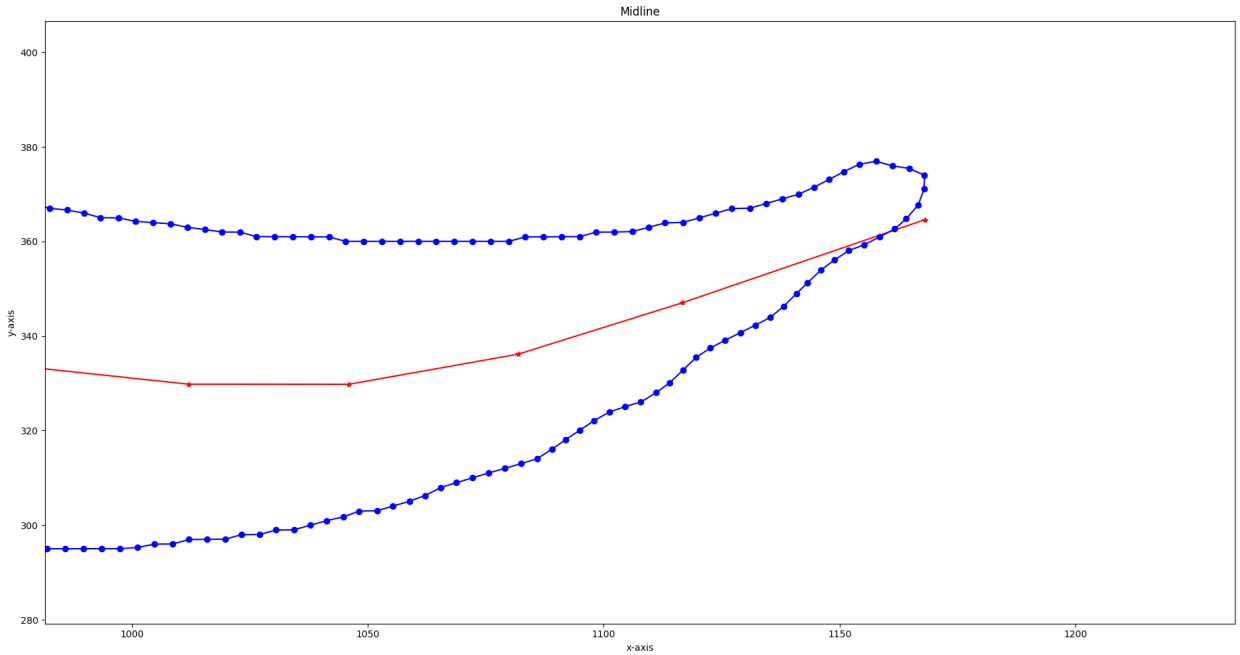


Figure 18: Example of how the ends of the fish are not always accounted for properly.

3.3 Verification

In this section an analysis is performed on the data gathered by utilizing the systematic approach detailed in the previous two sections of the results section. This is an important part of the process as it verifies that the results gotten from the the mask application and midline extraction are usable. The main point of this section is thus not to do an in-depth analysis of fish hydrodynamics for the salmon in the experiment detailed in the Material section, but rather to verify that the output of the method is usable. Had more time been available, a more in-depth analysis would have been performed, but the previous steps were heavily time-consuming.

3.3.1 Choosing data to analyse

To be able to do any sort of analysis, it is important to choose a meaningful dataset. As has been detailed in section 2, there were originally several fish that were in the SeaLab experiment, however one fish, fish 5, had clearer results than the others so only this fish was chosen. In addition to this, to get a clear result it is necessary to find a time frame where the fish is swimming as steadily as possible. Such a time-section was found, and it aligns with when the inflow velocity was at 1 BL/s. The chosen time-section was therefore from 10.15-10.20.

By importing the midlines it could be verified that the dataset was quite good. To further chose a dataset where the fish was swimming as close to absolutely steady as possible, the first 1.6 seconds were chosen. One way to glean a quick overview over the movement of the fish is to plot the position of the centroid in the x- and y-direction as a function of time. In figure 19 this movement can be seen. The centroid is simply calculated as the mean of every coordinate point in both the x- and y-direction. Both of these graphs indicate that there is significant movement in the dataset, so to get as clear a dataset as possible, the first 5 oscillations of the fish were chosen. This corresponds to frame 3 to 83, and a duration of 80/50=1.6 seconds. The whole dataset has in comparison 228 frames which corresponds to 4.56 seconds.

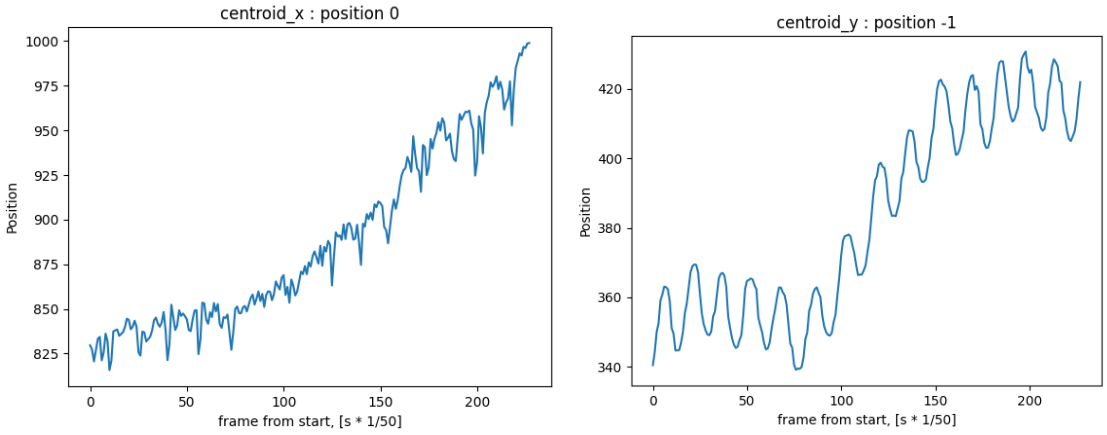


Figure 19: Movement of the centroid of the fish as a function of time. The graph to the left indicates the x-position of the centroid, and the one on the right indicates the y-position. Position 0 indicates the first point of the centroid vector, i.e. x-component, as does -1 indicate y-component

The now smaller dataset has been reviewed as an animated plot, however the format of a written

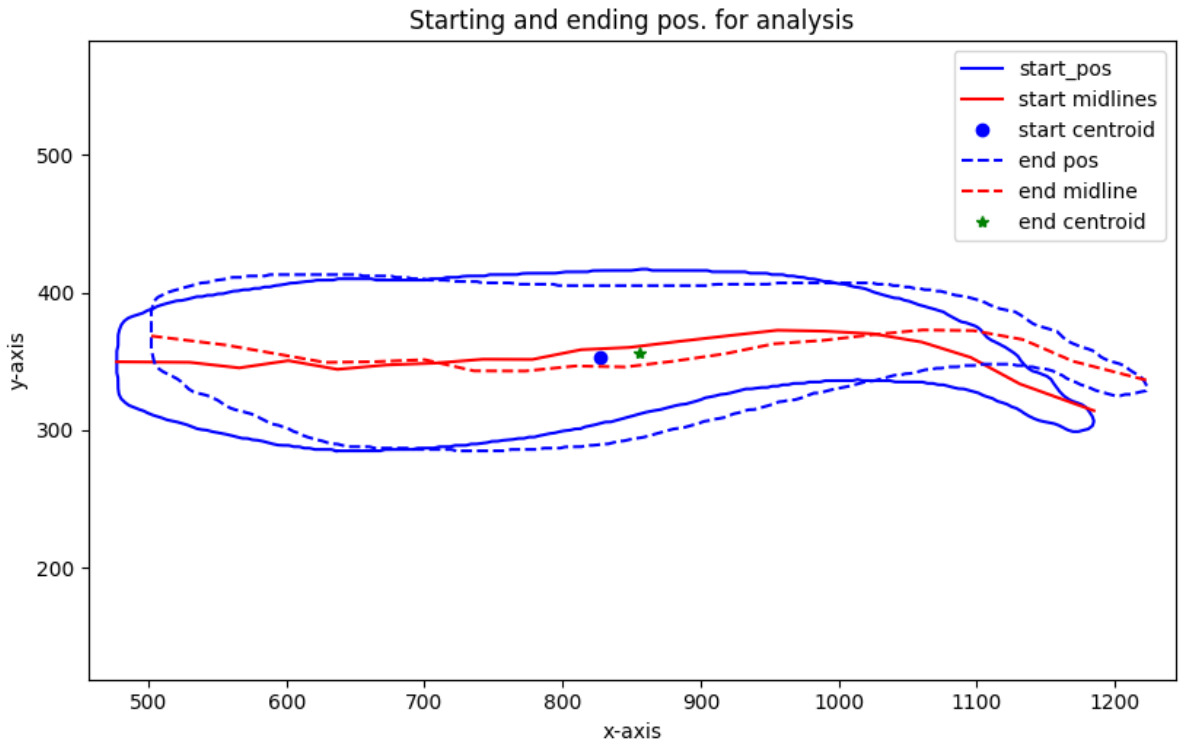


Figure 20: The position of the fish for the start and end of the chosen frames for the analysis. Start frame is frame 3 and end frame is 83.

thesis does not allow for easy verification by such means. However, the first and last frames of the dataset has been printed superimposed on each other in figure 20. In addition, the midlines and centroids are present. As can be clearly seen, the fish is almost in exactly the same position and shape, so this is deemed to be a reasonable dataset for carrying out further analysis. For the analysis, the y-coordinates of the midlines are more of interest. This is a simplification, but as the fish only does steady swimming in the chosen time frame, all lateral displacements are relatively small and as such the impact of including the x-direction can be neglected. To see then see how the fish moves over time, the y-position of every midline point is printed for each time instant in figure 21. As can be seen, there are 5 undulations and stays almost in the exact same spot. This corresponds well to what was seen in figure 19, but the centroid point does not necessarily reflect the whole midline so it is important to check.

Dataset quality

To start off the analysis of the data the quality will be assessed. The statistics the midline lengths over time are presented in table 4. There is certainly some deviation in the midline length, and specially in the midline sections. For the chosen time frame, there is less variation, signifying a better approximation of the fish in this section. The sources of this deviation are discussed in section 4, but they are most probably from detectron2 having trouble recognizing the entire tail region.

The fork length of the fish has been measured to be 29 cm, with this information, if we assume

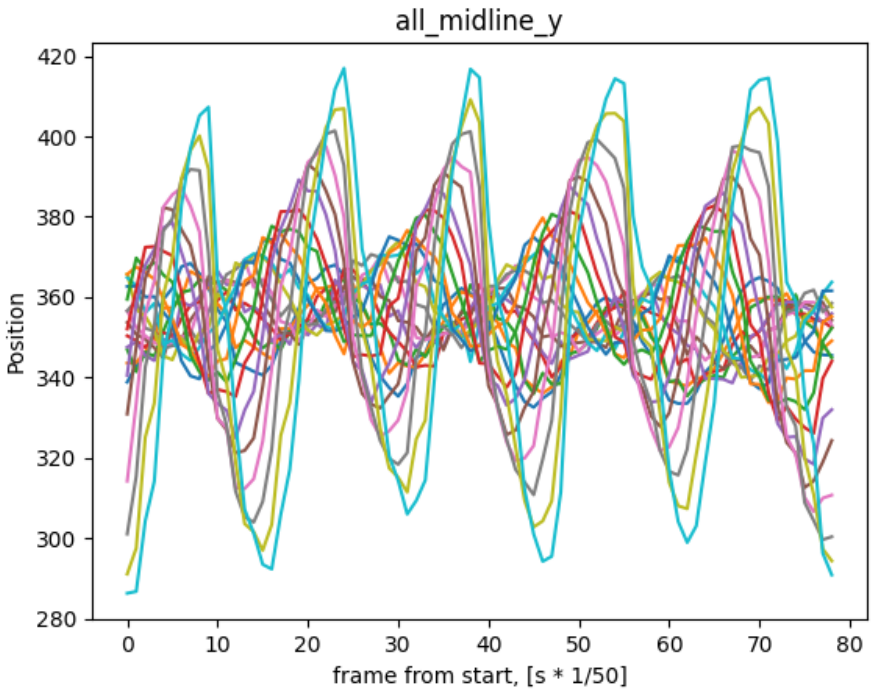


Figure 21: For each frame of video in the section, every midline point is plotted. This gives good insight into the motion of the fish, and a trend can be seen where the fish moves in a straight line with little deviation.

Table 4: The statistics of the length of the midline over time are presented below. Midline sections refers to the length between each individual midline point. Mean and Std are given as pixels, and Std(%) is given as percentage of the mean.

Property	Mean	Var	Var(%)
All frames			
Midline length	725.28	19.04	2.62
Midline sections	38.17	5.61	14.6
Frames 3-83			
Midline length	727.63	16.34	2.24
Midline sections	38.30	5.64	14.7

that the mean length of the midline is somewhat accurate, we can say that 725 pixels is 29 cm. If this is true, then the standard deviation of the midline length is around 0.76 cm, or 2.2% of the total length.

3.3.2 Analysis of data

The current midline has not been smoothed it is the pure output of the method described in section 3.2.2. It is will therefore contain the noise from any distortion in the actual image that the camera recorded in the lab, the noise from any imperfections in creating the detectron2 mask, and noise from the simple midline extraction method. One way to smooth out this noise would be to employ some sort of smoothing function over the midline. Another version is to use a Fourier transform to extract the frequencies of the midline motion and the remove the frequencies higher than the dominating frequency.

Fourier analysis

Using python, both the Fourier transform and the corresponding phase can be found by utilising the numpy library's fft and phase functions. The fft function uses the fast Fourier transform. By taking the fft of each midline point as a function of time, we can find the Fourier transform. Fourier analysis requires a periodic signal, but luckily the chosen dataset is now periodic with a period of 5, so there is no need to alter the input data. By processing the Fourier transform we can print the fft amplitude, presented in figure 22. The dominating frequency is found at 3.125 Hz for all midline points. This corresponds well with the data as this frequency multiplied with the duration of the signal gives 5 periods.

In figure 23 the Fourier signal has been scrubbed, meaning that all signals higher than the dominating frequency have been removed. This means that the resulting Fourier transform will look exactly like figure 22, except that from 5 Hz and up, the amplitude is 0. Whether the amplitude is correct can be quickly visually checked by seeing that the magnitude of the highest peak, which is the one corresponding to the tail end, is around 55, and that the amplitude of the oscillations of the same point, as seen in figure 23, also is around this number.

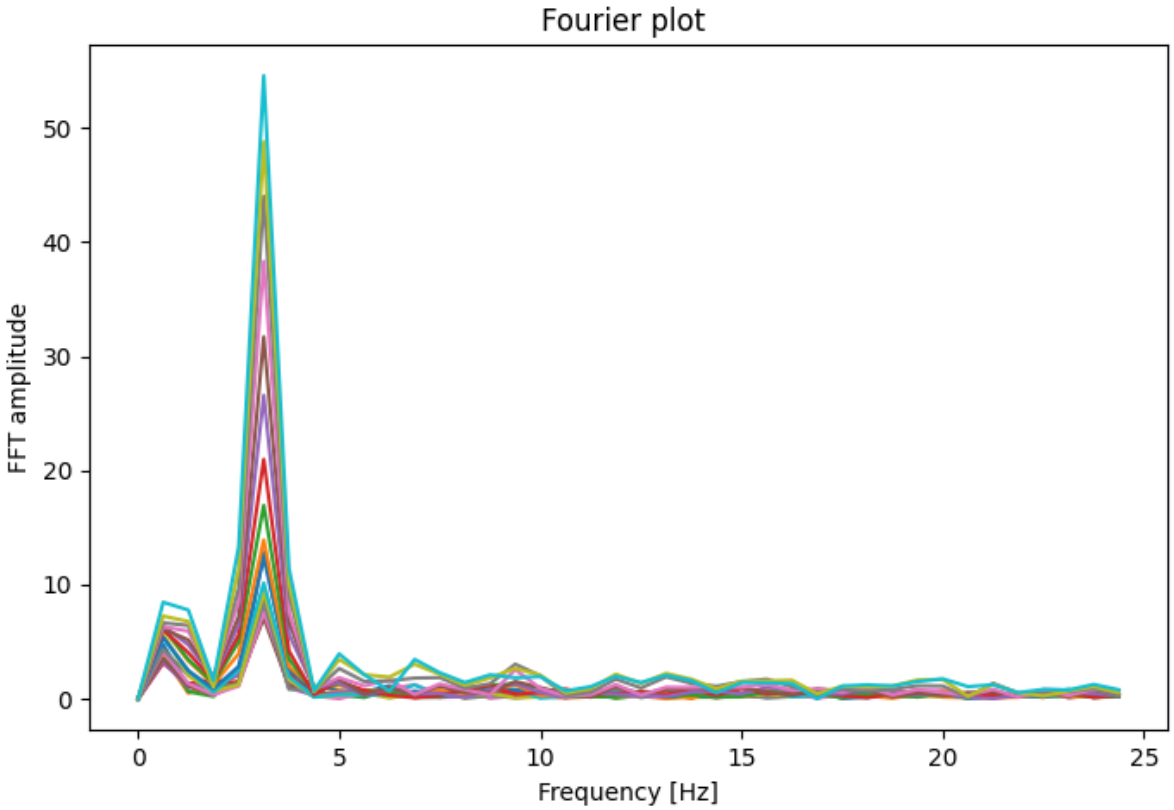


Figure 22: Amplitude of the fourier transform for the chosen dataset. There are two main peaks near origo, one indicating the slower periodic motion of the fish in sway, the other the dominating locomotion frequency.

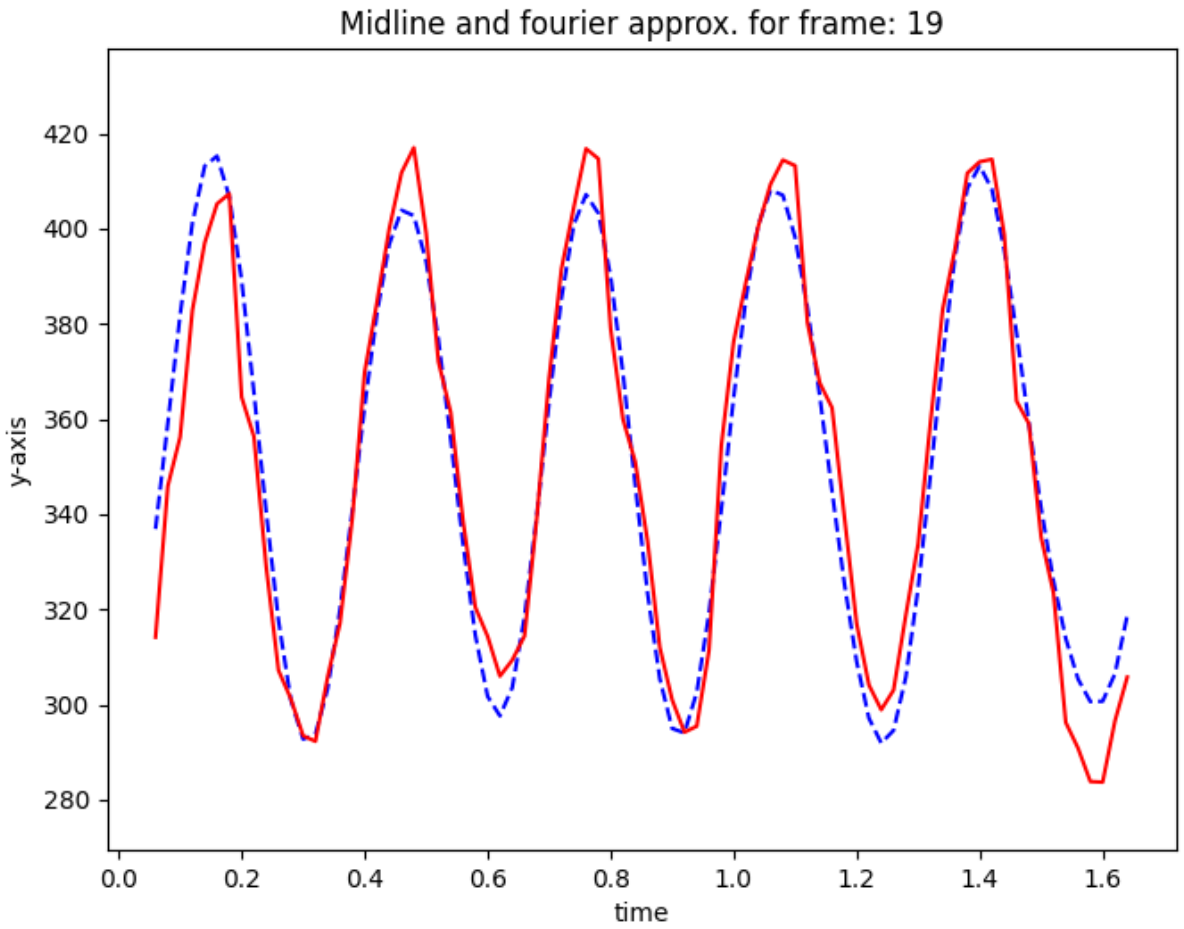


Figure 23: Fourier approximation of the last midline point, on the tail, overlaid the recorded position.

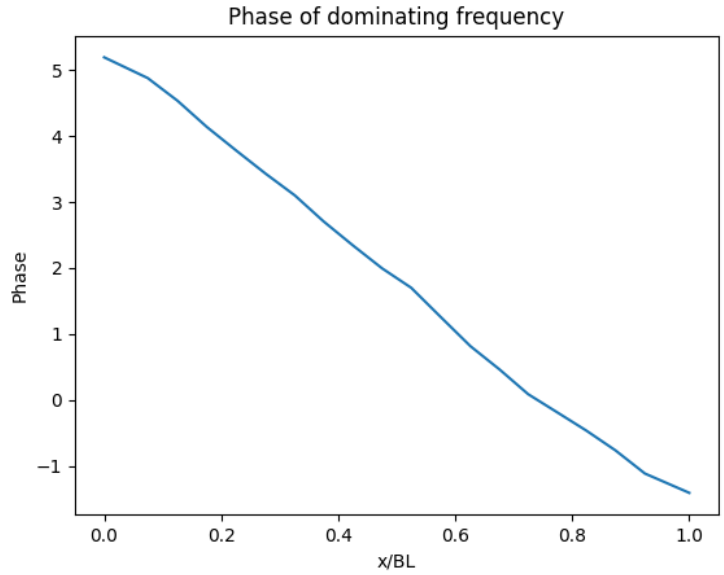


Figure 24: Phase of the dominating frequency as a function of midline x-position

Phase

The phase is found by computing the angle of the fourier transform. It correlates to the phase shift of the signal. In figure 24 it is seen that the phase of the dominating frequency is linearly shifting from 5.19 to -1.4, the difference is equal to 2π. Note that as the phase is circular as it comes from sinusoidal motion, there is a shift which is corrected by simply multiplying the first section by 2π. The actual values of the phase are only dependent on where the fish is in the motion.

Wave number and wave length

From the phase, the wave number is found, and likewise the wave length.

$$k = -\frac{d(phase)}{dx} \tag{20}$$

In our case, as the length of the fish is simply the unit length, the difference of the first and the last point in 24 are 6.602, which is 95% the value of 2π. Similarly, as they are inversely proportional, the wave length is found as

$$\lambda = \frac{2\pi}{k} \tag{21}$$

which gives a value of 0.95. This value is in relation to the body length, so it is 95% of the body length.

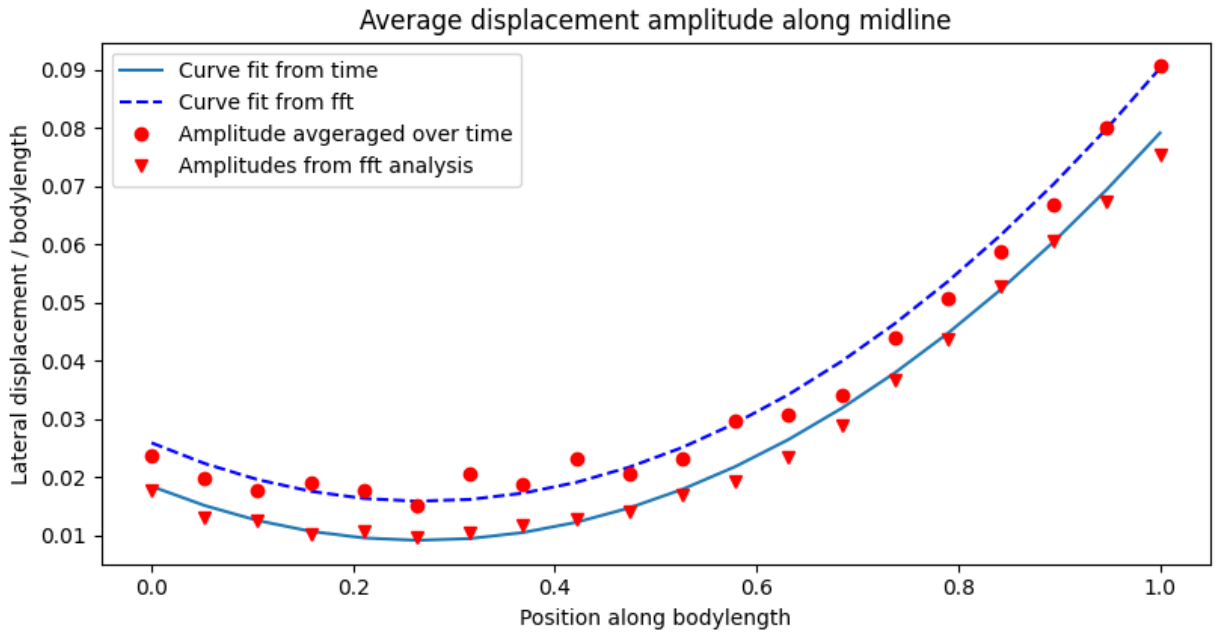


Figure 25: Amplitude envelope of the midline motion. Both estimations using time-analysis and fourier analysis have been used. Notice that the fourier estimation is significantly higher.

Amplitude envelope

The amplitude envelope says a great deal about the locomotion of the fish. As discussed in section 1.1.1 this envelope can be described by equation 19:

$$G(x) = \left(s_2 + \frac{-2s_4(s_3 - s_2)}{1 - 2s_4} x^* + \frac{(s_3 - s_2}{1 - 2s_4} x^{*2} \right) \tag{19}$$

The displacement envelope $G(x)$ has been found for the chosen dataset. It can be seen in figure **??** where the 20 midline points can be found as well as polynomial functions approximated in python.

Firstly, the method for finding the envelope by time-analysis was based on

```
for s in range(space):
    y_mean = np.mean(midlines_y[:,s])
    y_displacement[s] = (np.abs(np.max(midlines_y[:,s]))-y_mean)/mean_length
```

where space the for loop loops over each of the 20 midline points, mean_length is the mean midline length discussed above, and the amplitude envelope is found as y_displacement. For the fourier-based method, the envelope was found by

```
for s in range(space):
    max_y[s] = np.max(f_y)/mean_length
```

where f_y is the amplitude of the Fourier transform. From (Cui et al., 2018) we know that the parameters s2 and s3 can be read directly from the dimensionless amplitude envelope. This is

Table 5: Results from the analysis of the displacement envelope. The polynomials are of the form $p = ax^2 + bx + c$. For the parameters to the right, the values are listed under which method they were found by. All values are normalized in reference to the midline length. The left number for the travelling index is with all points averaged, the right is with all non-zero points averaged.

Method	a	b	c	s_2	s_3	s_4	k	λ	travelling index
Time	0.1389	-0.0746	0.0259	0.0259	0.09023	0.2684	-	-	0.688 / 0.725
Freq.	0.1301	-0.0694	0.0184	0.0184	0.07911	0.2666	6.602	0.951	-

achieved by calculating the polynomials for both the time and the frequency approximation for the head and the tail, to gain the values of s_2 and s_3, respectively. Now, by inspecting equation 19, we see that s_4 can be found easily by inspecting the term in front of the squared x. We see that

$$a = \frac{s_3 - s_2}{1 - 2s_4} \tag{22}$$

And since we already have a from the polynomial, we can now find s_4. The results are posted in table 5, where also the wave number k, wave length λ and the travelling index are listed. Note that all values are normalized in reference to the midline length.

The travelling index is found by the same method as outlined in 1.1.1. The analytic signal is found below. Notice that the mean of the midline points are subtracted from the to gain normalized values, as the hilbert function of numpy already produces the analytic signal.

```
for s in range(space):
    analytic_signal[:, s] = hilbert(midlines_y[:, s] - np.mean(midlines_y[:, s
                                                                ]))
```

From this the travelling index can be found as detailed in sectinon 1.1.1. By inspecting figure 26 one sees that a travelling index is actually found for each midline point, which alternates around a midline point. As presented in table 5 the mean of the travelling index is found as two different numbers depending on whether the first point should be counted or not.

Lastly, it is fitting to compare the findings of the current analysis with data from Cui et al., 2018, to see whether the results are reasonable. The found s_2 is squarely in the middle of the chosen species in table 6. For s_3, the values are actually outside what is found for the other subcarangiform species, but a better fit for carangiform species. s_4 is a good fit for all the entries. Lastly wave length and travelling index both agree with the other results.

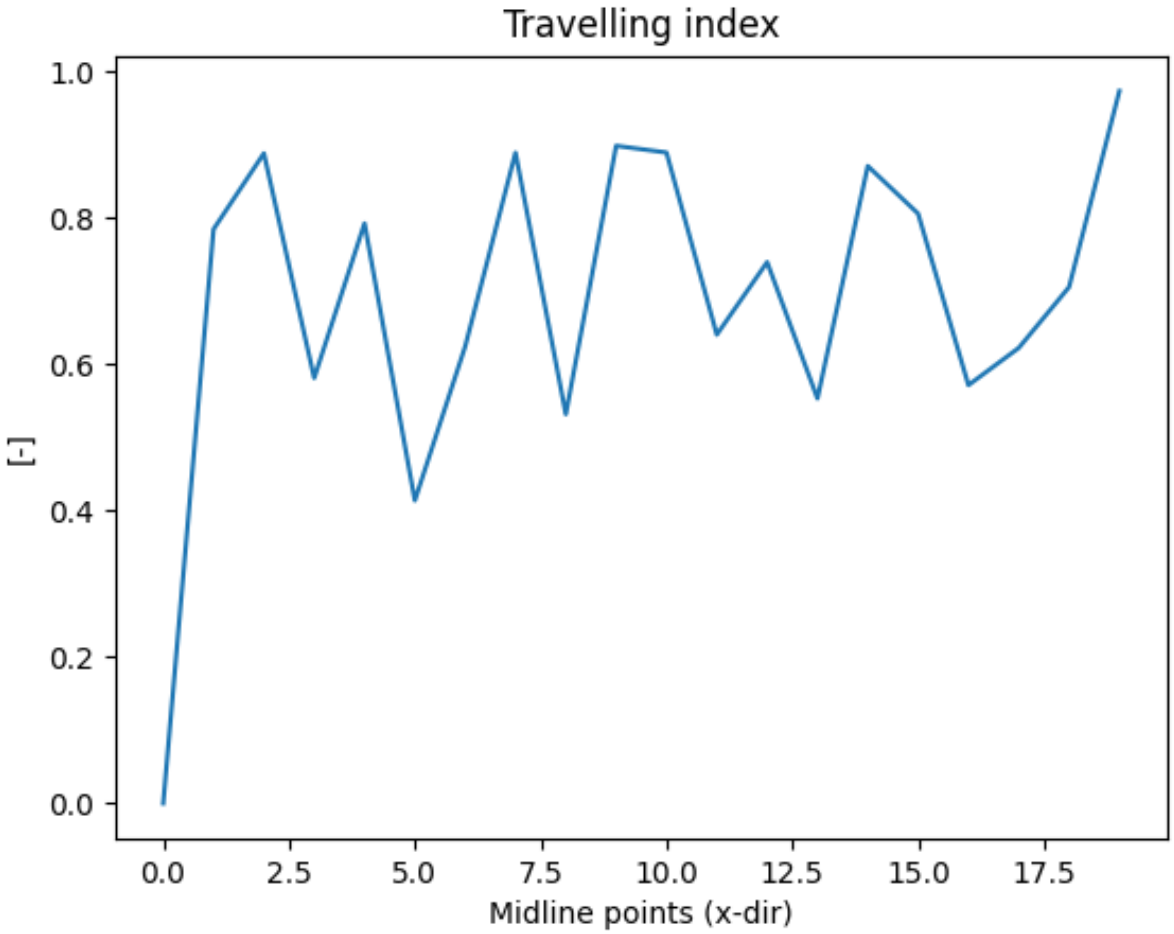


Figure 26: The travelling index of each midline point, from head to tail.

Table 6: A selection of characteristics from Cui et al., 2018. All units are dimensionless. s indicates a relative value which is independent of the traveling index.

Species	s_2	s_3	s_4	λ	travelling index
Subcarangiform species:					
Rainbow trout	0.0	0.12	0.0	0.7-1.3	0.61-0.77
Rainbow trout	0.156s	0.936	0.25	0.95	0.626
Rainbow trout	0.358s	0.994	0.2	0.9	0.777
Leopard shark	0.0	0.16-0.24	0.0	0.63-0.91	0.68-0.78
Zebrafish	0.0	0.275	0.2	1.0	0.665
Goldfish	0.058	0.174	0.337	0.882	0.623
Carangiform species:					
Largemouth bass	0.004	0.024	0.25	0.9	0.638
Largemouth bass Swim at 0.7 BL/s	0.0041	0.0472	0.3	0.590.83	0.580.68
Largemouth bass Swim at 1.2 BL/s	0.0053	0.0576	0.3	0.590.83	0.580.68
Saithe and mackerel	0.02	0.1	0.25	0.891.1	0.620.66

4 Discussion

There main goal of this thesis was to investigate a systematic approach to fish locomotion analysis.

4.1 Swimming experiment

There is some variation of the frame rate, but this should not be a problem as it is relatively fast. Still, the fish does move enough between each frame that a more detailed movement pattern would be documented had it been even higher. Importantly, the images are not blurred and the motion is captured sufficiently.

There are several factors that limit the usefulness of the experiment. However, as the main goal of this thesis is to create a systematic approach to salmon locomotion extraction, it is enough to simply be able to analyze the resulting midline to verify that the approach is valuable and can be used in further studies.

If fish locomotion studies are to be done in the future, there are certain aspects that should be improved. For one, the size of the tank relative to the fish should be larger, as in the current setup there are many instances where the fish is swimming close to the surface of the tank. This creates surface effects which interfere with the pure, unaffected inflow which is sought. In addition to this, the curved glass does not offer any advantages in terms of optical distortions. One might also wonder how the flow Reynolds number affects the fish locomotion. As salmon naturally travel in both rivers and the open ocean, it can reasonably be assumed that tight, turbulent spaces are not an unnatural habitat for the fish, but a glass or plastic tank is not a natural environment and will undoubtedly induce stress in addition to interfering with the pure, unaffected steady locomotion that is sought.

4.2 Mask identification

Deep learning is in large part treated as a black box in this thesis. Deep learning is a complex subject and the tools required to wield it are difficult to understand, install and run. Even so, quite good results were achieved by someone with limited knowledge. The major weakness in the implementation was that the details of the model used to build on was unknown, the variables in the configuration of the model were unknown, and the dataset trained on was limited. Even so, with under 30 annotated images to train on, the results were usable. This is very promising to the future of mahcine learning implementations of fish locomotion.

The post-processing part of the process is all in all not a significant contributor to noise. Although the smoothing function of the mask was not perfect, the improvements gained by improving the way the model predicted the mask points are much more significant. The fact that the masks were completely lacking points in large areas, and quite consistently misrepresented both the tail and the head region means that improvements here would yield vastly better results.

Other types of deep learning networks are also worth to investigate, however the benefits of sticking to detectron2 is that a working implementation has now been made, and improving this might give a better yield for less work. There are also versions of detectron2 that simply build

on it, but use the same framework. There are also different networks that specialize in different things, such as sharp edges for the tail.

4.2.1 MASB

If complex movements are ever to be analysed, then future implementations of the masbpy algorithm are a viable options. There were many problems with the masbpy implementation. The masbcpp version, which uses C++ was not able to run at all. As this is the newest version, it might be that it actually will work except for some simple bug. At any rate, converting the masbpy framework from python2 to python3, and then having to rewrite most of the functions was highly time consuming and should have been dropped earlier in the process. The idea of having a much more robust tool was what made dropping the method harder.

The main problem lies in the normal vectors rarely working perfectly, as well as the shrinking balls connecting to the neighbouring points and thus giving wrong results. One might wonder if much better performance might be had if detectron2 had given better output. This is hard to know for certain, but might be unlikely as masbpy did not perform well even with few coordinate points to work on. The example videos show masbpy and masbcpp working very well even with large datasets, which makes it perplexing that it did not work even for something that should have been relatively simple.

In the end, other methods should be investigated before this is attempted again. The method might give good results but hopefully better alternatives exist.

4.2.2 Simple approximation

The main disadvantage to this method it is not suitable to scale to more advanced methods of fish locomotion analysis. This method is linked to the limitations of the experiment it is based on, which for our case is fine enough, but it would have been better to make the MASBY method work as it is a more robust mehtod and would be possible to implement in future cases also.

One disadvantage of this method is how it locks the user into only analysing fish aligned along the x-direction. For the specifics of this thesis, this was not a problem and as such

4.2.3 Verification

Very large variance. This can is probably from the inaccuracy of the midline predictions of the tail. 0.76 cm st. deviation of midline length probably affects the results, and it should be better. This is a deviation of almost 3%, for a truly accurate analysis this should be improved.

Phase: The fact that the line is not completely straight is probably due to noise, although it could technically also be due to the fish.

4.3 Further works

As knowledge of bio-inspired robotics improves, it is reasonable to want to study body interactions with varying flow fields, such as swimming around objects, navigating through obstacles or

interacting with other swimmers (the task is left to the reader). I believe that filming fish swimming in the wild would provide very good research material. Although this might be difficult to implement in the near future, larger tanks with more space for the fish to swim in are definitely possible. Also, rectangular tanks allowing for maneuvering would also be highly beneficial for study of locomotion which is to include steering. In addition, a higher resolution camera with an even better frame rate would also be better for getting as accurate results as possible for the deep learning algorithm.

The detectron2 algorithm must be improved if better results are to be had. There are many aspects that could be improved, such as has been identified in the discussion previously. However, for proper results, it would be best to involve someone with a proper understanding of machine learning. There is no substitute for experience, and if detectron2 were to be fine-tuned it will most probably deliver better, faster results.

The only thing remaining is for others to take the system into use, so that it can be optimized for further data extraction.

5 Conclusion

The goals of this thesis were 1. to summarize the findings of the project thesis, 2. to describe the performed fish swimming experiments, 3. create a working environment in python using MLAs and apply it to the experiments, 4. to attempt a validation of the method created in step 3, and finally 5. to draw conclusions from the studies.

The first two goals have been achieved in the first section of the thesis.

The main goal is goal 3. It proved to be a massive undertaking to create an environment which could utilize MLAs for fish locomotion analysis. This is because deep learning is a demanding subject and the tools used to wield it are difficult to master. Due to the nature of framework of detectron2, a robust system had to be made, so that the amounts of data could be transferred from one step in the process to the next. If this information pipeline was not robust, there would have been no point in trying to automate the process to begin with.

The systematic approach has proved to be successful, as it is now relatively easy to get as much data as one would want. Due to time concerns, not much time was left to further analyse the data to gleam something meaningful into the experiments, but since the systematic approach actually works, now other can. There are certainly many points of improvements to the systematic approach, such as investigating further implementation of detectron2; improved training data and altered configuration setup, as well as possibly augmenting detectron2 with newer versions built upon it.

Goal 4 could have been implemented better, such as by using the least squares method to separate recoil from the transverse locomotion. But given the circumstances it served its purpose to show that the results from the systematic approach are usable and not garbage.

All in all the goals of the thesis have been adequately met. The systematic approach produced results and they were verified against other data. Now all that remains is for the approach to be utilised to produce usable result in more detailed research.

Bibliography

Breder, C (Jan. 1926). "The locomotion of fishes". In: *Zoologica*. URL: https://scholarworks.umass.edu/fishpassage_journal_articles/359.

Cui, Z. et al. (Apr. 2018). "Complex modal analysis of the movements of swimming fish propelled by body and/or caudal fin". In: *Wave Motion* 78, pp. 83–97. ISSN: 01652125. DOI: 10.1016/J.WAVEMOTI.2018.01.001.

Detectron2 (n.d.). *Detectron2 documentation*. URL: https://detectron2.readthedocs.io/en/latest/.

Garbade, Michael (2018). *Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences*. URL: https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb.

Géron, Aurélien (2017). "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow (2019, O'reilly)". In: *Hands-On Machine Learning with R*, p. 510.

He, Kaiming et al. (n.d.). *Deep Residual Learning for Image Recognition*. Tech. rep. URL: http://image-net.org/challenges/LSVRC/2015/.

Lin, Tsung-Yi et al. (2015). "Microsoft COCO: Common Objects in Context". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3686–3693. ISSN: 10636919. URL: http://arxiv.org/abs/1405.0312.

Ma, Jaehwan et al. (2011). "3D medial axis point approximation using nearest neighbors and the normal field". In: DOI: 10.1007/s00371-011-0594-7.

Maertens, Audrey P., Amy Gao, and Michael S. Triantafyllou (2017). "Optimal undulatory swimming for a single fish-like body and for a pair of interactingÂ swimmers". In: *Journal of Fluid Mechanics* 813, pp. 301–345. ISSN: 14697645. DOI: 10.1017/jfm.2016.845.

Microsoft (2019). *Visual Object Tagging Tool: An electron app for building end to end Object Detection Models from Images and Videos*. URL: https://github.com/microsoft/VoTT.

Moen, Frode (June 2020). "Experimental Analysis of Sea-bass Hydrodynamics". PhD thesis. Trondheim: NTNU.

Peters, Ravi (2014). *Medial Axis - Shrinking Ball Algorithm on Vimeo*. URL: https://vimeo.com/84859998.

Peters, Ravi and Hugo Ledoux (2014). "masbpy". In: *Github repository*. URL: https://github.com/tudelft3d/masbpy.

Peters, Ravi, Kevin Wieve, and Jeff Coukell (2015). *tudelft3D/masbcpp*. URL: https://github.com/tudelft3d/masbcpp.

Ptak, Bartosz (n.d.). *GitHub - UAVVaste/VoTT2COCO: Script that converts VoTT json files to COCO format. Keeps images, bboxes, and masks*. URL: https://github.com/UAVVaste/VoTT2COCO.

scipy (n.d.). *Scipy docs*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html.

Sfakiotakis, Michael, David M. Lane, and J. Bruce C. Davies (Apr. 1999). "Review of fish swimming modes for aquatic locomotion". In: *IEEE Journal of Oceanic Engineering* 24.2, pp. 237–252. ISSN: 03649059. DOI: 10.1109/48.757275.

Xu, Huili (2019). *Swim tunnel observation experiment*. Tech. rep.

Yuxin Wu et al. (2019). *Detectron2*. URL: https://github.com/facebookresearch/detectron2.

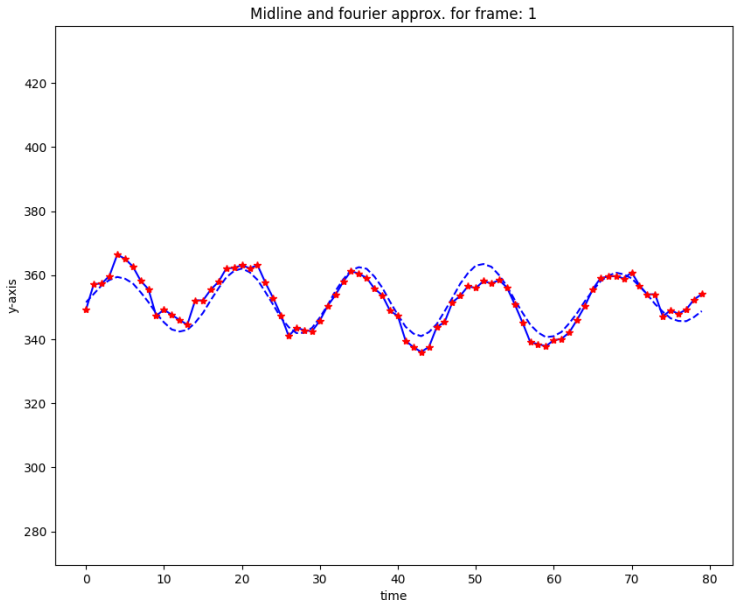


Figure 27: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 1 of 19.

Part I

Appendix

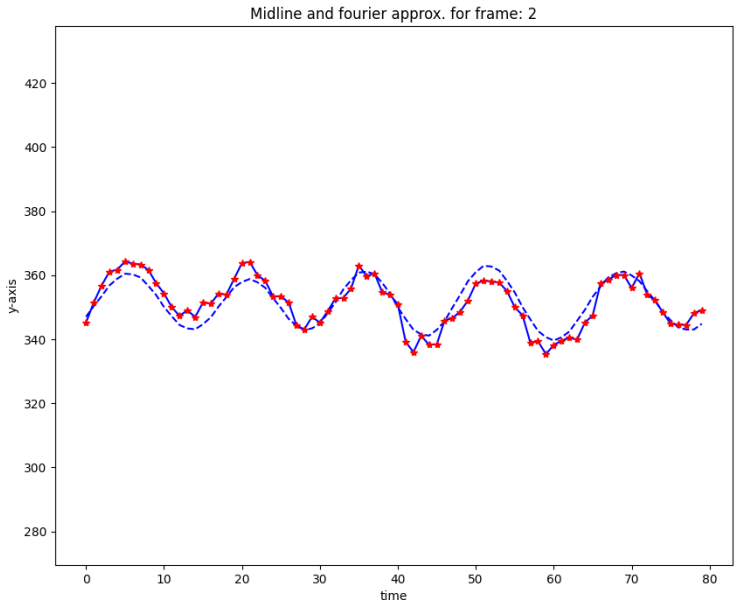


Figure 28: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 2 of 19.

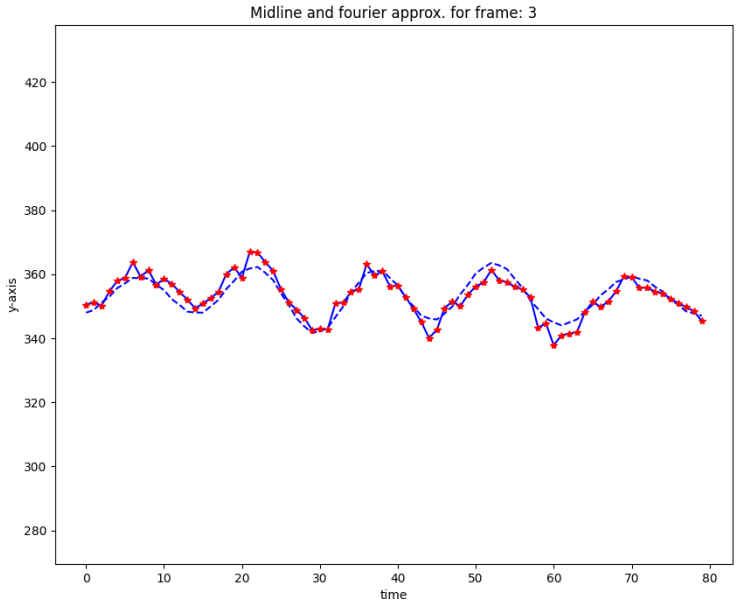


Figure 29: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 3 of 19.

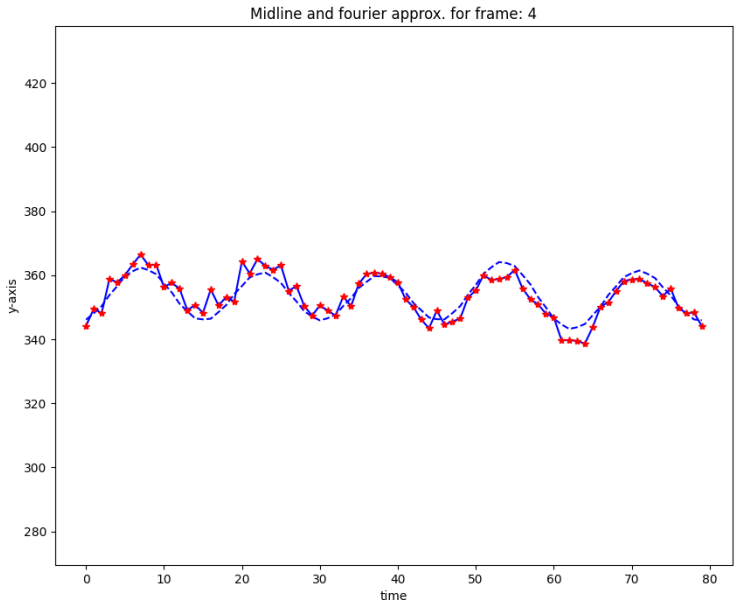


Figure 30: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 4 of 19.

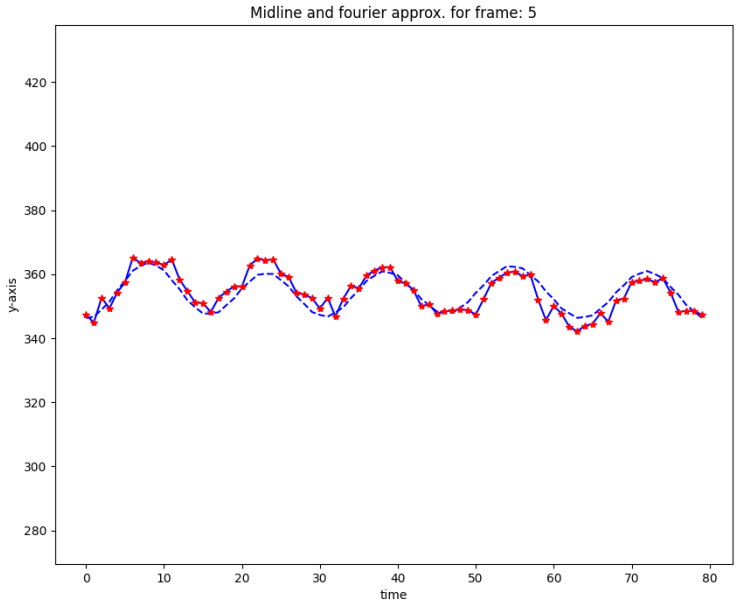


Figure 31: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 5 of 19.

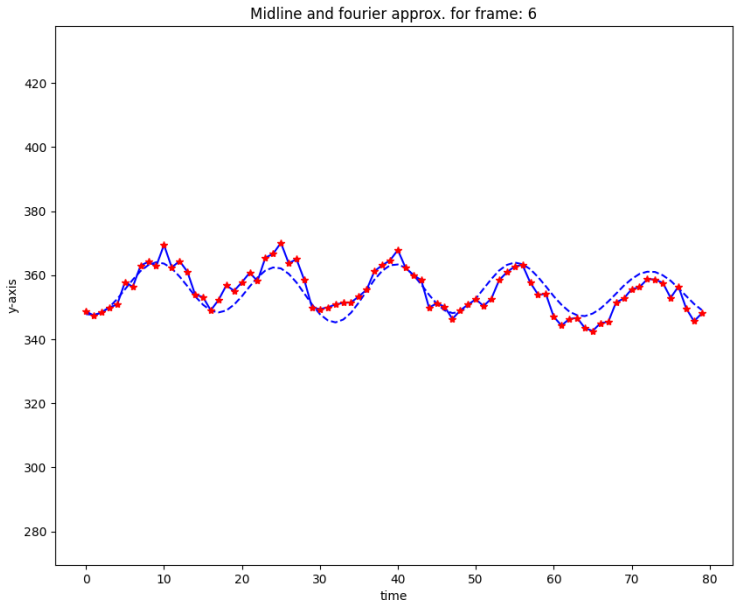


Figure 32: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 6 of 19.

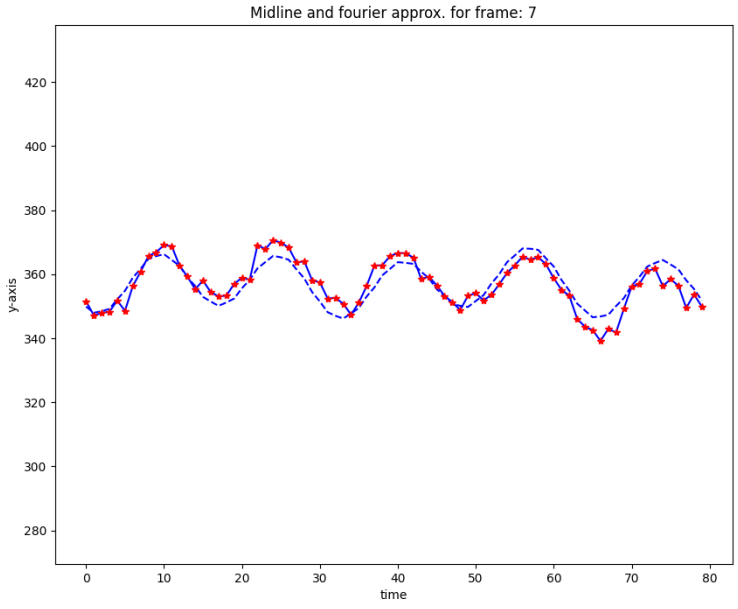


Figure 33: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 7 of 19.

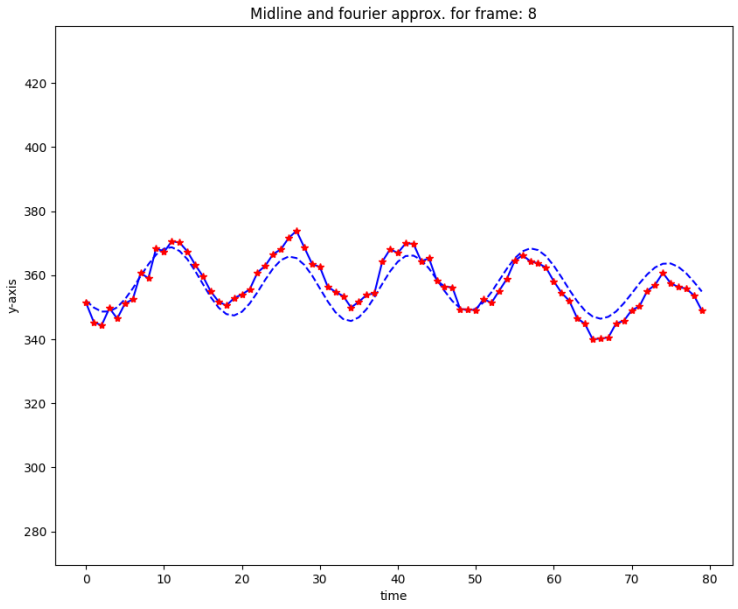


Figure 34: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 8 of 19.

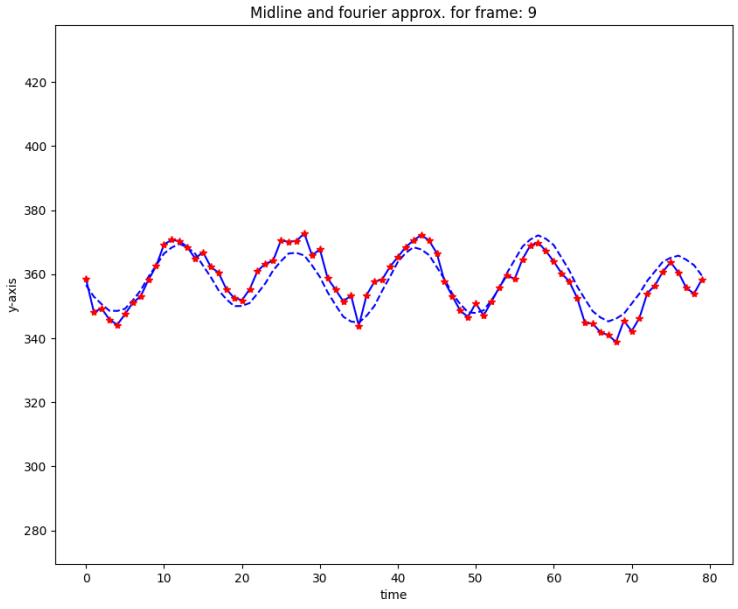


Figure 35: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 9 of 19.

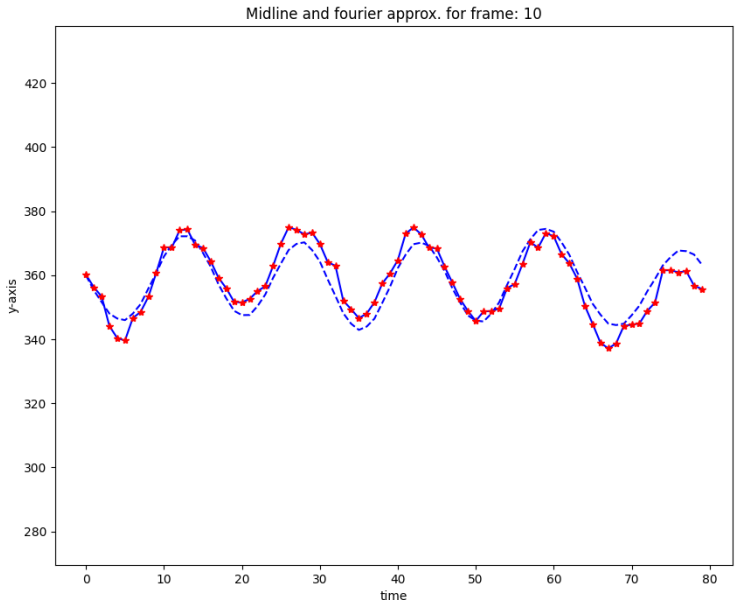


Figure 36: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 10 of 19.

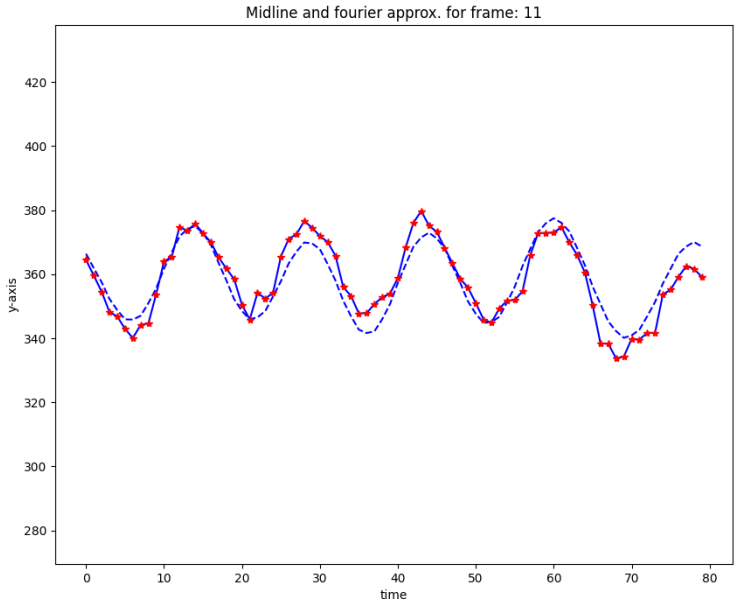


Figure 37: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 11 of 19.

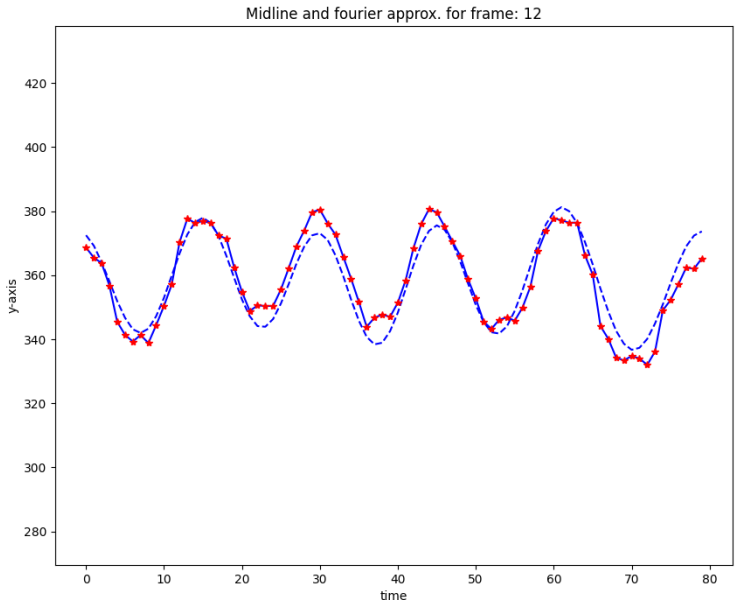


Figure 38: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 12 of 19.

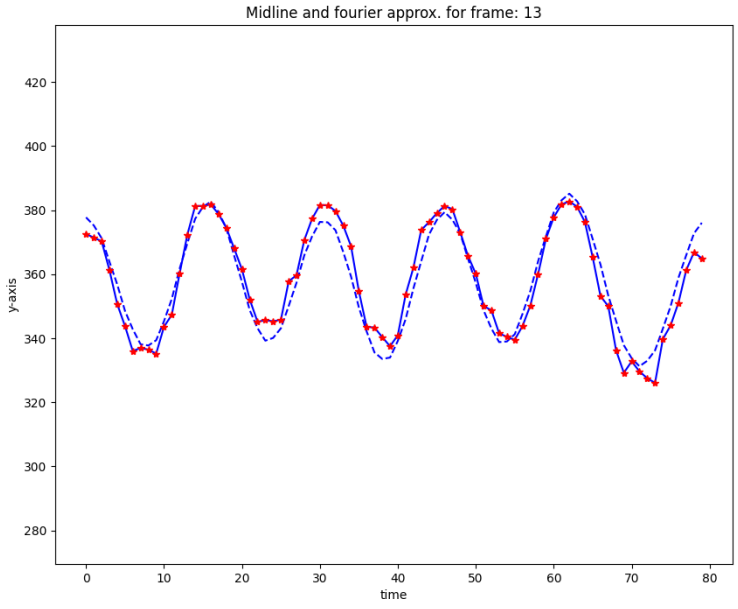


Figure 39: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 13 of 19.

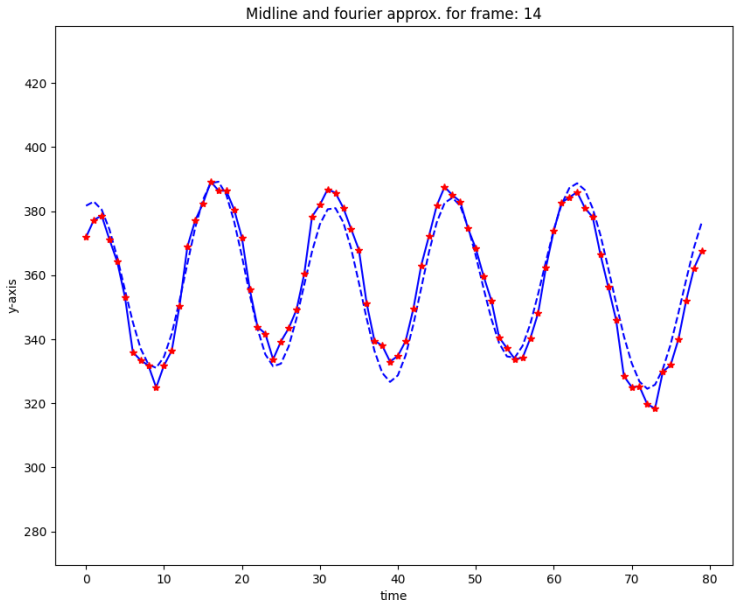


Figure 40: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 14 of 19.

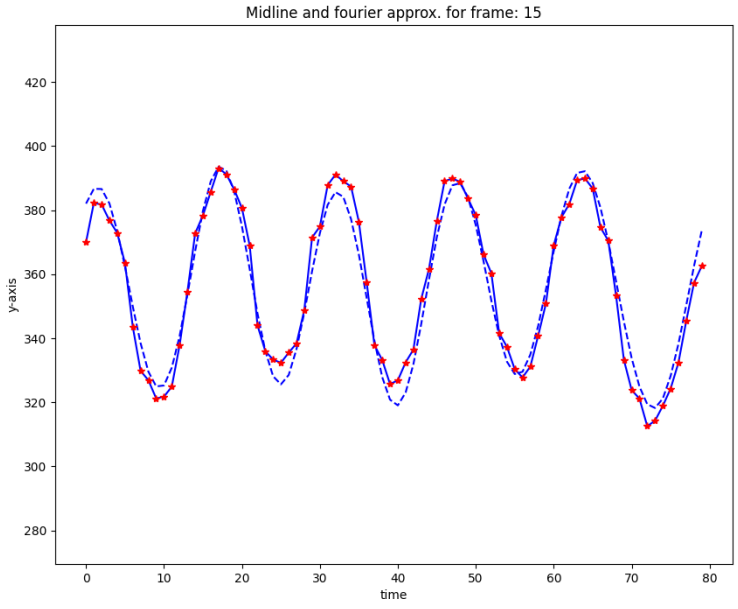


Figure 41: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 15 of 19.

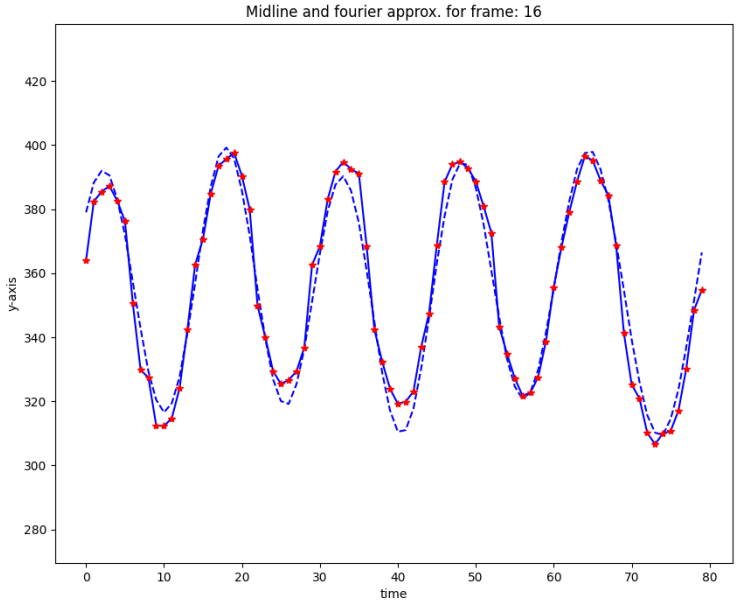


Figure 42: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 16 of 19.

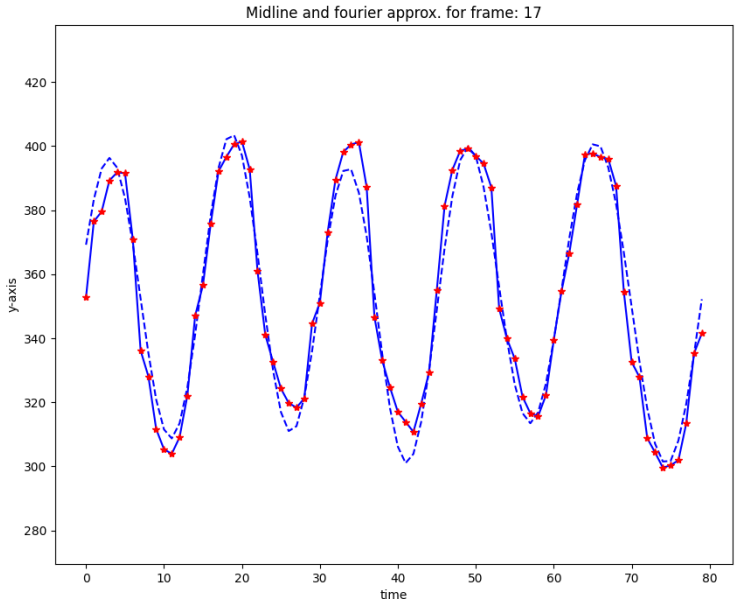


Figure 43: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 17 of 19.

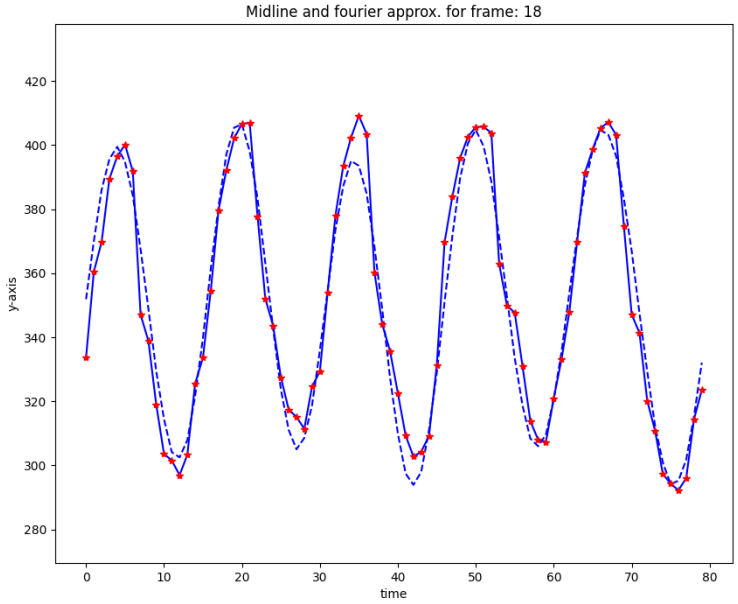


Figure 44: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 18 of 19.

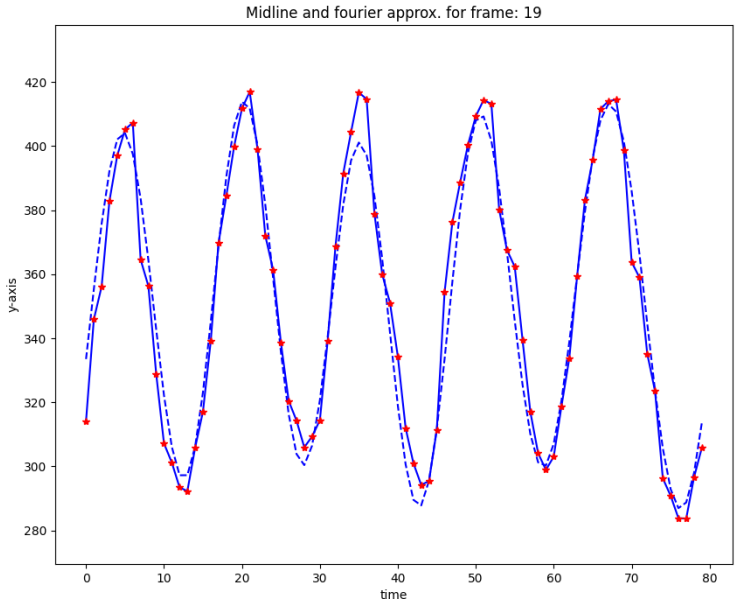
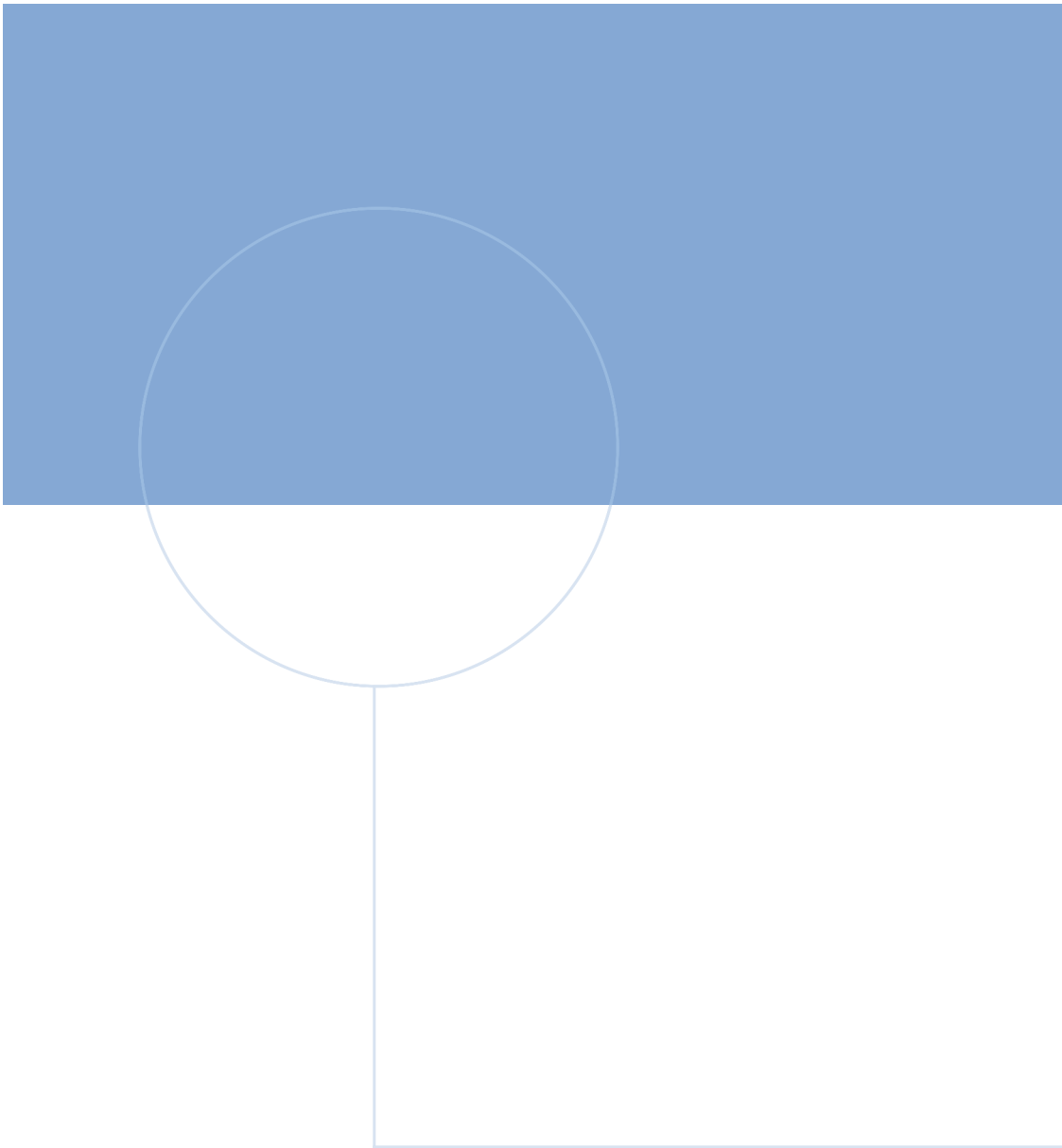


Figure 45: Midline y-displacement and fourier approximation using only the dominating frequency of 3.125Hz for position 19 of 19.



NTNU

Kunnskap for en bedre verden