Detection, Tracking and Length Estimation of Free-Swimming Fish with a Range-Gated Camera

Ingrid Utseth

Thesis submitted for the degree of
Master in Computational Science
60 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2020

Detection, Tracking and Length Estimation of Free-Swimming Fish with a Range-Gated Camera

Ingrid Utseth

Detection, Tracking and Length Estimation of Free-Swimming Fish with a Range-Gated Camera

http://www.duo.uio.no/

Abstract

With a global decline in fish stocks and a growing demand for seafood, there is a need for non-intrusive technological systems that can improve automatic data collection for conservation efforts and economic efficiency in the fishing industry. Previous studies have utilized computer vision and machine learning techniques to analyse data captured by underwater cameras and extract information regarding the size and species of the fish present in the images. Underwater footage is often of poor quality, and challenging to analyse due to the high variability in the fish's shape and intensity.

SINTEF has developed an innovative range-gated camera system capable of 3D estimation underwater. This thesis presents an algorithm which utilizes the depth map produced by this system to detect, track and estimate the lengths of the fish observed by the camera.

Multiple segmentation techniques, based on global and local thresholding, edge detection and background subtraction, were applied to 93 hand-segmented frames and evaluated using four validation metrics. A highly customizable tracking algorithm based on a minimum-cost flow network was developed, capable of tracking the fish through occlusions and missed detections. Using a dataset with three known fish lengths, ways of alleviating length estimation errors due to inaccuracies in the depth map or the segmentation were explored. The tracker ensures that the length estimates can be refined using multiple observations of the same fish, increasing the likelihood of the fish being observed in an optimal position for length estimation.

The final algorithm was able to accurately detect and track all fish appearing in a test set consisting of 501 frames with three free-swimming codfish, using Euclidean distance to link detections and the estimated velocity direction of the fish to resolve occlusions. The average percentage error in the length estimation was approximately 10%.

These results are promising, but more data with known fish lengths is needed in order to thoroughly analyse the length estimation algorithm. The algorithm was applied to a dataset containing schooling fish with unknown lengths; in this case the fish were only partially tracked due to a high number of occlusions and the velocity directions of the fish being largely the same.

Contents

List of Symbols

I	Image
t	Time / Frame number
c	Speed of light
z	Distance from camera
(X, Y, Z)	Point in the real world coordinate system [cm]
f	Camera focal length
T	Threshold
x	Horizontal pixel position
y	Vertical pixel position
G	Number of graylevels in a grayscale image

Preface

This thesis marks the end of two years as a student in the master's programme Computational Science: Imaging and Biomedical Computing. It was written in collaboration with SINTEF. This project has been frustrating at times, but more often fascinating.

I would like to thank my supervisors, Fritz Albregtsen at the Department of Informatics (IFI) at the University of Oslo (UiO) and Petter Risholm at SINTEF. I have received outstanding support, advice and guidance throughout the development of my algorithm and the writing process. I greatly appreciate the biweekly meetings with Petter (extremely helpful and motivating whenever I got stuck) and Fritz' experience and willingness to repeatedly read and comment on my writing.

Moving back home due to a global pandemic was not part of my original plan. I would like to thank my parents and sister for nice dinners, a new exercise regime, good company and endless support during the uncertain, difficult and stressful last few months of my degree. I am also grateful to my friends for giving me good advice, keeping me motivated and cheering me up with long video calls. Special thanks go to Iris Anna Berg and Henrik Gjestang. My two years at UiO would have been a lot less enjoyable without you. I look forward to dinners, new projects, climbing, travelling and the next "Paint 'n Sip."

Part I

Introduction

Chapter 1

Introduction

The aim of this thesis is to develop an algorithm that can reliably detect and track the movement of free-swimming fish captured in 3D by a range-gated underwater camera, and use this information to return an accurate length estimate for each observed fish. A successful algorithm is a step towards a system that can provide efficient and non-intrusive monitoring of the biomass, health and species of the fish present in the monitored area.

1.1 Background and motivation

The collapse of the cod stocks off the coast of Newfoundland, Canada, serves as a grim warning of the possible consequences of overexploitation. In 1992, the cod biomass dropped to one percent of its previous levels. [18] A moratorium was announced in the summer of 1992; initially intended as a short-term solution, it is still largely in effect. The collapse affected some 39000 jobs [40] and by 2015, the total spawning cod biomass in the region measured only about a third of the biomass prior to the collapse. [63]

The sea provides economic stability and food security to hundreds of millions of people, but the current exploitation of marine species is unsustainable. [15] From 1974 to 2015, the portion of the world's fish stocks that are overfished rose from 10% to 33%. The lowest percentage of sustainable fish stocks are found in the Mediterranean and Black Sea region; 62% of the fish stocks in this area are depleted due to overfishing. [70] Simultaneously, global seafood consumption per person doubled between 1960 and 2016, from about 10 kg to 20 kg, and it is expected to continue to rise. [15]

In addition to depleting stocks of the target fish species, commercial fishing can cause great harm to non-target species. 40% of all global marine catches is estimated to be bycatch, incidentally caught non-target fish, discarded at sea or at port due to wrong species, size or quality. [12] One example would be the hundreds of thousands of pelagic dolphins killed in purse seines, surrounding walls of nets drawn closed at the bottom underneath schools of fish, in the 1960s and 70s. [33] New netting technologies along with trade restrictions have greatly reduced the dolphin mortality, but the population is still not recovered. [33]

With the global decline of fish stocks and a growing demand for seafood, the development and implementation of sustainable fishing practices is paramount. SMARTFISH H2020 is a four-year long, EU-funded project which aims to develop and test new technological systems that improve automatic data collection. Accurate and easily available data can increase economic efficiency, reduce unintended fish mortality and ensure compliance with regulations. [46]

This thesis was written in collaboration with SINTEF, one of several partners in this project. All data analysed in this thesis stems from tests and demonstrations of a range-gated underwater camera system developed by SINTEF. Such a system can, for instance, provide real-time monitoring of the interior of a trawl to ensure no unwanted species are caught. [64]

1.2 Goal and method

While the main objective of this thesis is to produce accurate length estimates for all fish detected by the camera in a certain time period, the success of the length estimation is dependent on reliable detection and tracking of the fish. Accurate detection, or segmentation, ensures that the fish length is not constantly underestimated due to parts of the fish missing. With a reliable tracking algorithm, the length estimate can be refined using multiple observations of the same fish.

To find a segmentation method capable of detecting fish extremities such as fins and tails in addition to the main body, several popular image segmentation methods were evaluated using a number of validation metrics. The non-uniform lighting, low contrast and the variability in the shape of a fish depending on its position and orientation with regards to the camera make segmentation difficult, but by utilizing the 3D-information from the range-gated camera, these challenges can be overcome.

The tracker will be based on a minimum-cost flow network, with a specially adapted cost function developed to maximize the probability of linking two observations belonging to the same fish. Several possible cost functions will be evaluated using multiple tracking validation metrics. The tracker should be capable of keeping a fish's unique identification number even in challenging cases of missed detections or occlusions over several frames.

Accurate length estimation is difficult due to the contortions of the fish as it swims or its orientation with regards to the camera, which may result in some parts of the fish not being visible in the frame. The final length estimate is therefore based on several detections of the same fish, increasing the likelihood of the fish being observed in an optimal position for length estimation. Possible sources of errors as well as ways to minimize these will be investigated. The success of the length estimation will be evaluated by comparing the estimated length to the true length of the fish.

1.3 Scope

This thesis is based on data captured by a range-gated underwater camera system developed by SINTEF. This system is still under development, with limited data available. While it has been tested in the sea, the data from these particular tests had too much noise and too few fish sightings to be used in this thesis. Instead, this thesis only considers data collected in pools under far better conditions than one can expect to find in the sea. Another difference to note is that we observe the fish from the side in these datasets; when the camera is affixed to a trawl, we would observe the fish from above.

The results in this thesis should only be treated as "proof-of-concept" as they have yet be rigorously tested on data collected under more realistic conditions; most of this thesis is based on a single dataset. The evaluation of the final estimated lengths is based on the measured lengths of three codfish. Naturally, three data points are not sufficient to properly estimate the final error.

The peak-finding algorithm used to generate high-resolution depth maps from the raw sensor data was developed by SINTEF. It is described in Section 3 and in further detail in Risholm et al. [54]

1.4 Contributions

The final algorithm is intended to be a step towards a more complex data collection system for trawling, able to estimate the biomass of the fish currently in the trawl. With meticulous testing of how various popular segmentation methods perform on this type of data, this thesis identified several possible methods that return results with high accuracy and minimal segmentation noise.

The tracking algorithm outlined in this thesis performs well on the main dataset analysed; it was capable of correctly tracking the fish in the test set through occlusions involving all three fish. The algorithm is not expected to be successful on datasets with a lot of fish in each frame, but it will likely be able to return some partial results. The algorithm can also easily be adapted to other types of datasets.

The estimated percentage error of the length estimation algorithm was around 10%. These are promising results, but keep in mind that with only three known fish lengths in datasets analysed, rigorous error evaluation of the length estimation was not possible. More testing is necessary. It is however clear that basing the length estimate on multiple observations were necessary as the length estimates vastly improved for well-positioned fish.

1.5 Thesis structure

Part II, Theory and Methodology, is divided into several chapters. First, previous work related to fish detection and tracking is presented in Chapter

2. Then, the range-gated camera system and the process of extracting real world coordinates for length estimation is described in Chapter 3. Chapter 4 presents the data produced by the camera, and describes the formats of the validation datasets. Then follows an overview of the proposed pipeline in Chapter 5, before methods that may be used in the segmentation and tracking parts of it are described in chapters 6 (segmentation) and 7 (tracking) before fish length estimation is described in Chapter 8. Finally, an overview of the metrics used to validate the results is presented in Chapter 9.

While results and discussion are typically presented in different chapters, this thesis presents results and their related discussions together in part III, Experimental Results and Discussion, for ease of reading. Part III, chapter 10 presents all experimental results and is divided into four sections; the first three concerns segmentation, tracking and length estimation on the main dataset studied in this thesis, while the fourth section demonstrates the algorithm's performance on a different dataset. Part III, Chapter 11 summarizes the results in Chapter 10.

Finally, part IV contains a conclusion and suggestions for further work in this field.

Part II

Theory and Methodology

Chapter 2

Related work

A common way to gather data regarding the average weight and size distribution of a fish population (highly important information for both conservation efforts and in aquaculture) is to capture and measure a small sample. This manual procedure is time-consuming, often inaccurate, and may cause physical harm to the fish. [34] Efforts have therefore been made to develop more accurate and non-invasive biomass measurement methods, often utilizing computer vision and machine learning techniques to analyse the data captured by underwater cameras. A few selected approaches are outlined in the following section. Li et al. [34] offers a more comprehensive review of biomass estimation techniques for aquaculture specifically.

Analysing underwater footage comes with certain challenges. Underwater images are often low contrast with non-uniform lighting. The fish themselves display a high variability in intensity and shape. Therefore, most methods have only been successfully tested in controlled environments [39], such as tanks or narrow channels where the background is uniform and the fish is likely to be captured from the side. Such data collection methods are not harmful to the fish, and certainly applicable to the fish farming industry, but they are difficult to adapt to data collected in the sea with less uniform backgrounds, likely more turbulent water and more random motion of the fish. There is still need for a system capable of collecting and analysing data from, for instance, a trawl.

Fish detection and segmentation

Tillett et al. [67] used a 3D point distribution model (PDM) to estimate dimensions of free-swimming fish. The model was fitted to 3D images collected in a tank with stereo cameras and achieved an average error in length estimation of 5%. However, the model required manual initialization of the points and was very sensitive to the initial point placement. The authors noted the difficulty in analysing images of salmon in regular production cages, as opposed to the tank, due to light scattering in the water, the natural camouflage colouring of the salmon and a non-uniform background which may occlude the fish.

Lines et al. [36] took steps towards a system for estimating the biomass

of free-swimming salmon in sea cages. Once again, the problem of the low difference in contrast between the fish and the background was noted. An n-tuple binary classifier was used to identify the characteristic crescent shape of a fish head when subtracting an image slightly separated in time from each frame. Points are identified automatically along the fish body using a PDM similar to Tillett et al. and the mass was estimated from linear measurements. The mean mass measurement error was 18% after preliminary tests on 60 frames with 17 fish.

Zion et al. [75] analysed images of fish swimming, one by one, through a narrow channel with background light. Segmentation by subtracting an image with no fish produced a clear contour of the fish. Object contour signatures were extracted and used to classify three fish species.

Chuang et al. [9] describes an algorithm for automatic segmentation of fish sampled by a trawl-based underwater camera system. The algorithm first finds positions of probable fish using local thresholding. The initial segmentation is then further refined. The final method achieved a 78% recall against ground truth on low-contrast underwater images.

Atienza-Vanacloig et al. [3] also tested their method on underwater images acquired under real conditions. Their fully automatic method segments the fish using background subtraction or local thresholding (which they remark is fast and behaves well when the background is not uniform). After segmentation, a deformable fish model is automatically adjusted to the fish shape and size. The model achieved a 90% success rate on videos captured in natural conditions. The authors note that overlapping fish is a problem that still needs to be resolved.

Multiple-object tracking

Qian et al. [50] presents a method for tracking multiple zebrafish which was adapted to be used in this thesis. The fish were dark against a static white background, making segmentation by background modelling fairly simple. The cost function is based on fish head position and direction, optimized between consecutive frames. Basing the cost function on fish heads minimized occlusions, as the model could still be updated when other parts of the fish were obscured.

Network flows as used by Qian et al. have previously been used to solve general tracking problems, as the cost function can easily be adapted to different situations. Zhang et al. [74] adapted a network flow-based optimization method to track multiple pedestrians. The algorithm was expanded with occlusion hypotheses, added to the observation set in instances where an object was considered likely to be directly occluded by another object. The algorithm showed improvement compared to previous results on the same dataset, and the authors note that the framework can be easily adapted to tracking any class of objects.

Henriques et al. [21] applied a similar network flow-based optimization method to a pedestrian tracking problem. Contrary to Zhang et al., this method pre-linked detections yielding a set of tracklets (track segments). The tracklets were then linked using an optimal matching method, in this

case the Hungarian method, to form the final set of tracks. Occlusions were accounted for by allowing merges and splits to connect to virtual termination and initialization nodes respectively.

Chapter 3

Range-gated camera system

The range-gated camera system developed by SINTEF delivers real-time 3D estimation underwater, based on a peak determination algorithm that is robust to scattering. The camera system consists of a camera with a black-and-white CMOS chip and a 542 nm solid-state laser with a repetition rate of 1 kHz. It is discussed in more detail in Risholm et al. [54]

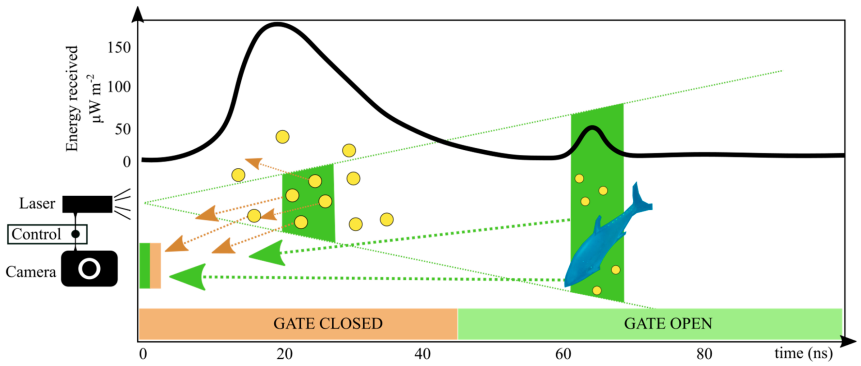


Figure 3.1: The graph shows the reflected signal $I'(x,y,z)$ from a laser pulse. By gating out the first 50 ns (marked in orange) most of the reflected signal from particles in the water (backscatter) is removed from the image. The camera shutter is only open for a short time window, capturing objects at distance increments Δ_z from the camera. Figure from [38].

Let $I'(x,y,z)$ represent the number of photons collected at pixel (x,y) from an infinitesimal range at distance z. Integrating I' from a distance z and beyond produces an image $I(x,y,z)$ gated at distance z (see Figure 3.1).

$$I(x,y,z) = \int_z^\infty I'(x,y,z)dz \tag{3.1}$$

With gating delay steps of $\Delta_t = 1.67$ ns and the speed of light underwater being $c_w \approx 22.5$ cm/ns, the system has a spatial sample increment of $\Delta_z = \frac{1}{2}\Delta_t c_w = 18.8$ cm. For each distance $z = z_{min} + \Delta_z i, i = 1, ..., N_r$, an average image $\hat{I}(x,y,z)$ is calculated from N_a images from the same distance. Increasing the number of images used in the averaging reduces

noise, but N_a is constrained by the transfer rate between the camera and the PC used for processing and visualization. [54]

In environments with no scattering, the best depth estimate will be the range from which the most number of photons are reflected; meaning that the highest peak in the differentiated delay sweep curve at pixel (x, y), $I'(x, y, z)$, corresponds to the distance to the target. To be used underwater, the system needs to handle the effects of light scattering and introducing false peaks in the signal. As shown in Figure 3.1, the peaks due to forward and backwards scatter may be higher than the signal peak due to the actual target. [54]

The effects of scatter can also be seen in Figure 3.2. The object in the turquoise square is 7 m from the camera. The peak at 3.2 m in the differentiated delay sweep curve is forward scatter from the target to the right, ranged at 3.2 m. This peak is higher than the peak at 7 m due to attenuation of the signal with distance. Backscatter from particles close to the camera causes the rise in the signal from 2 m to 0 m.

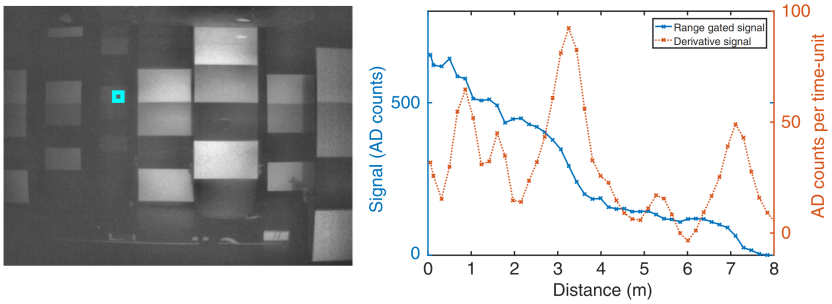


Figure 3.2: The histogram-equalized intensity image gated at 2 m from the camera (left). Right shows the effects of scattering on the negative differentiated delay sweep curve (dashed red) from the turquoise square. The blue graph is the signal. The object in the turquoise square is 7 m from the camera, but peaks due to backscatter close to the camera and forward scatter from the target to the right at 3.2 m are higher than the peak due to the actual target. Figure from [54].

The algorithm takes the effects of scattering into account by selecting the most distant peak higher than a noise floor T_n. A good threshold was selected through simulations. The derivation is done by convolving the signal with a derivative kernel combined with a smoothing filter in the z-direction. The smoothing filter reduces the effects of noise, which is increased in the derivation.

Higher resolution was achieved through interpolation of the discrete signal. Noting that the peak of the curve resembles a parabola, a parabolic fit based on three central points was used in the interpolation to find the center position. Experimental results showed that the parabolic fit was more robust than a weighted average with more points, due to the average being biased with regards to the selection of the points. With signal interpolation and the parabolic fit, resolution was increased from $\Delta z = 18.8$ cm to 0.8 cm.

The signal-to-noise ratio was further improved by binning the images before computing the depth map. After computing the depth map, uncertain depth estimates are removed. The peak height is used as an uncertainty metric. Laser intensity noise is reduced in the foreground by averaging intensity images gated close to the camera. Similarly, sensor noise is reduced by subtracting an averaged background image, an image gated at a distance far away from the camera, from the foreground.

3.1 Real-world coordinates from the depth map

The real world coordinates (X, Y, Z) of a point in the image can be calculated from the angle of view (AOV) and the point's distance to the camera z. AOV, illustrated in Figure 3.3, describes the part of a 360-degree circle (horizontal or vertical) that is captured by the sensor. [13, p. 121] Knowing the focal length f and the dimensions of the sensor in millimetres, the total horizontal and vertical AOV can be calculated as follows: [69]

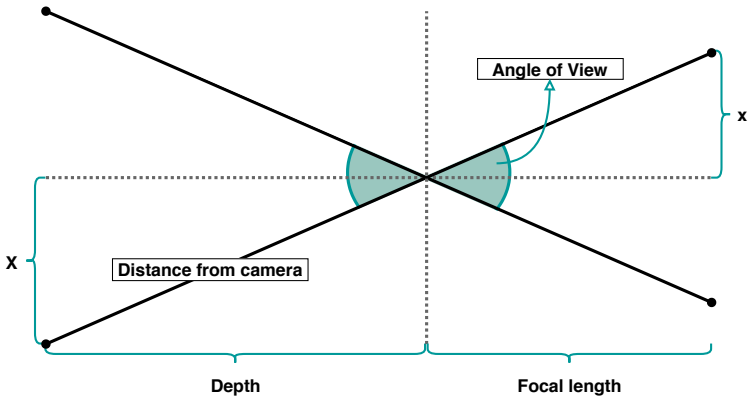


Figure 3.3: 2D Pinhole camera model illustrating AOV for a pixel at position x. Using x, AOV and distance from camera z, the real world coordinates (X, Z), where Z is the object's depth, can be calculated.

$$AOV_h = \frac{1}{\eta} 2tan^{-1}(\frac{w}{2f}) \tag{3.2}$$

and

$$AOV_v = \frac{1}{\eta} 2tan^{-1}(\frac{h}{2f}) \tag{3.3}$$

w and h represent the horizontal and vertical dimensions of the sensor respectively (width and height). The sensor dimensions are 960×512 pixels, and each pixel is $14\mu m$. Thus w = 13.44 mm and h = 7.17 mm. $\eta \approx 1.32$ represents the refractive index of air/water. [53]

In order to maximize the accuracy of the calculated coordinates, the AOV of each individual pixel is calculated, rather than simply dividing the

total AOV by the horizontal or vertical number of pixels. Consider a pixel at position (x, y). Let $\hat{x}$ and $\hat{y}$ represent that pixel's horizontal and vertical distance to the image center. The horizontal and vertical AOV at pixel (x, y) is calculated:

$$AOV_{\hat{x}} = \frac{1}{\eta} 2tan^{-1}(\frac{\hat{x} * \rho}{f}) \tag{3.4}$$

and

$$AOV_{\hat{y}} = \frac{1}{\eta} 2tan^{-1}(\frac{\hat{y} * \rho}{f}). \tag{3.5}$$

Here, $\rho = 14\mu m$ represents the size of each pixel. The change in horizontal AOV based on pixel distance from image center is illustrated in Figure 3.4.

Knowing the AOV and z, the point's distance to the camera, the horizontal and vertical real world coordinates may be calculated as follows:

$$X = sin(\frac{AOV_{\hat{x}}}{2}) \cdot z \tag{3.6}$$

$$Y = sin(\frac{AOV_{\hat{y}}}{2}) \cdot z \tag{3.7}$$

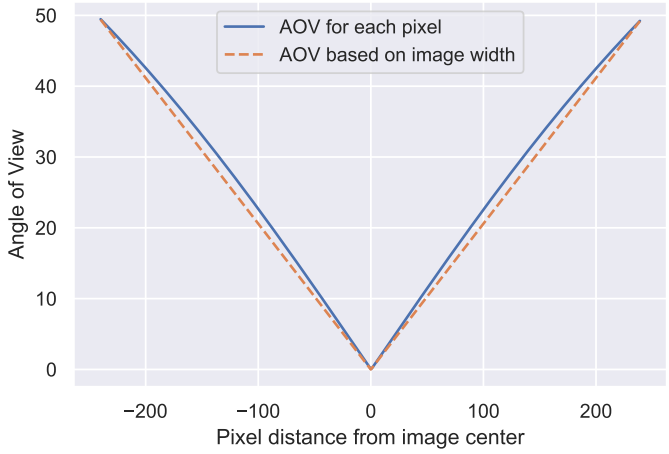


Figure 3.4: Comparison of the horizontal AOV calculated for every pixel, and AOV calculated based on the image width. Note that non-linear increase in the AOV calculated for every pixel; the difference in AOV between neighbouring pixels is larger near the image center than near the edges of the image.

Finally, the real world depth coordinate Z is easily calculated using Pythagoras theorem:

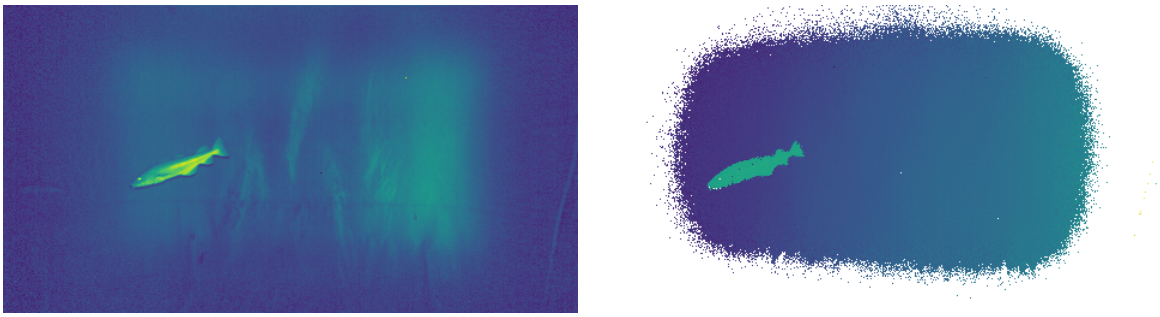
$$Z = -\sqrt{z^2 - X^2 - Y^2} \tag{3.8}$$

Chapter 4

Datasets

4.1 3 Cod

The main dataset used in this thesis was filmed in a pool and includes three free-swimming cod. The lengths of the fish were manually measured to be 71 cm, 54 cm and 41 cm. Note that the conditions are likely better than we can expect to find in the sea, with less noise due to the water being less turbulent and a more uniform background.



(a) Intensity frame 39

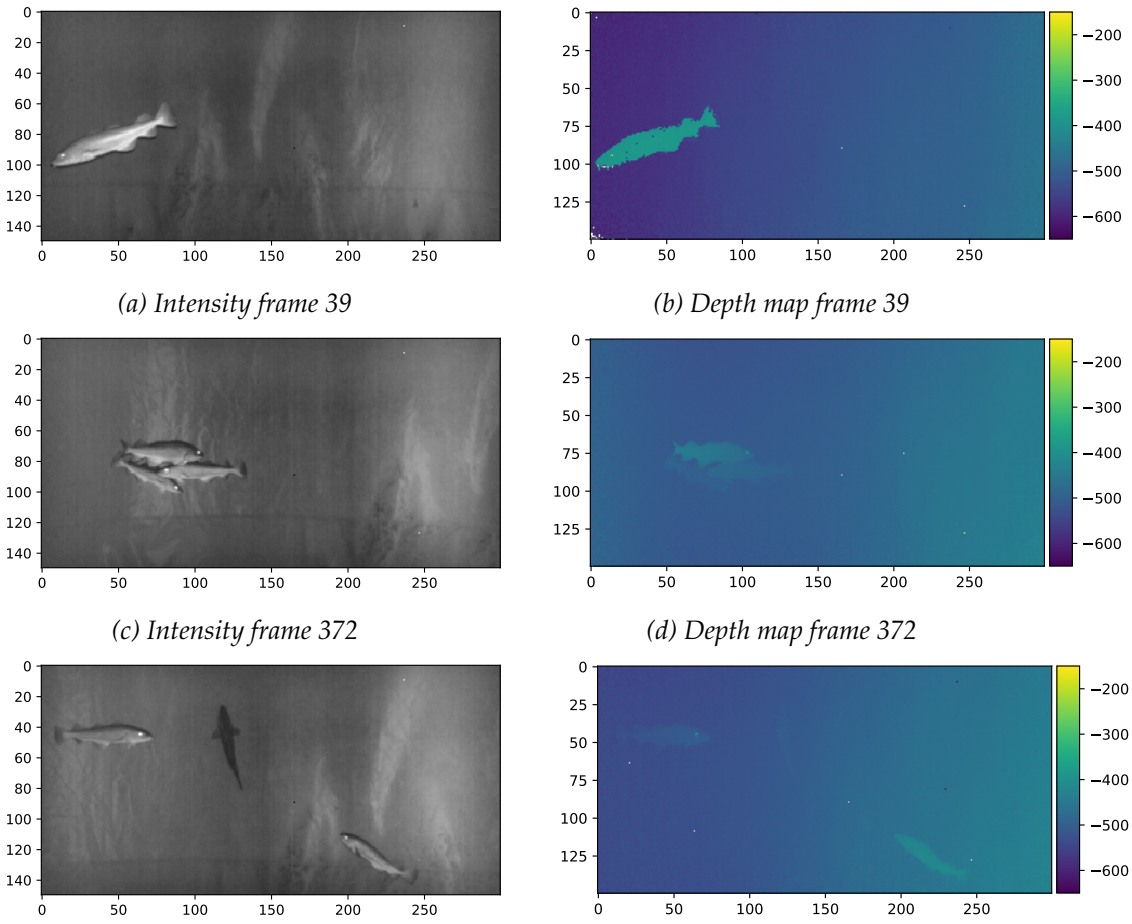
(b) Depth map frame 39

Figure 4.1: Example images from the 3 Cod dataset. The white parts of the depth map indicate None-entries; pixels for which the depth could not be reliably estimated.

As Figure 4.1 illustrates, the depth map contains a lot of None-entries, mainly located around the edges of the image, due to high uncertainties in the distance calculations. Since very little information could be extracted from these areas, each frame was trimmed from the original size of 480 $\times$ 256 to 300 $\times$ 150 (see Figure 4.2).

4.1.1 Validation sets

Two validation sets were created from the 3 Cod dataset; one to measure the accuracy of the segmentation and one to measure the accuracy of the tracking. The segmentation dataset is a set of 93 images (approximately every 20th frame in the video) with manually drawn binary masks, as



(a) Intensity frame 39

(b) Depth map frame 39

(c) Intensity frame 372

(d) Depth map frame 372

(e) Intensity frame 626

(f) Depth map frame 626

Figure 4.2: Cropped example images from the 3 cod dataset. Most of the None-entries from the depth map have been removed. The depth map illustrates the fish's and the background's distance to the camera in cm.

shown in Figure 4.3. The fish were not individually labelled, as the goal is to measure how successful the initial segmentation of the fish and background was.

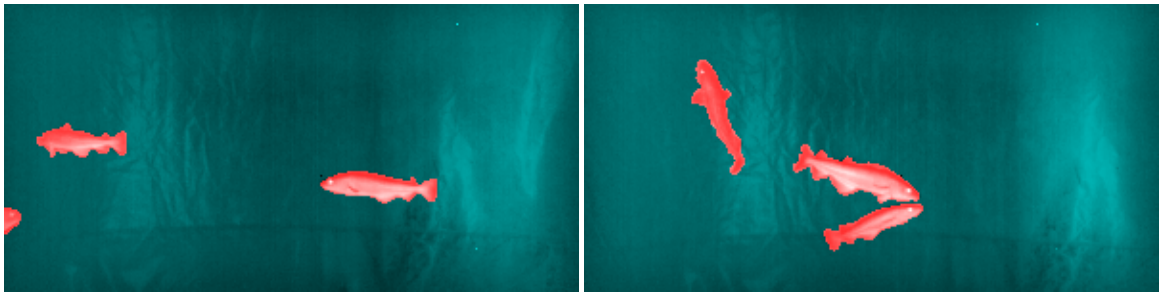
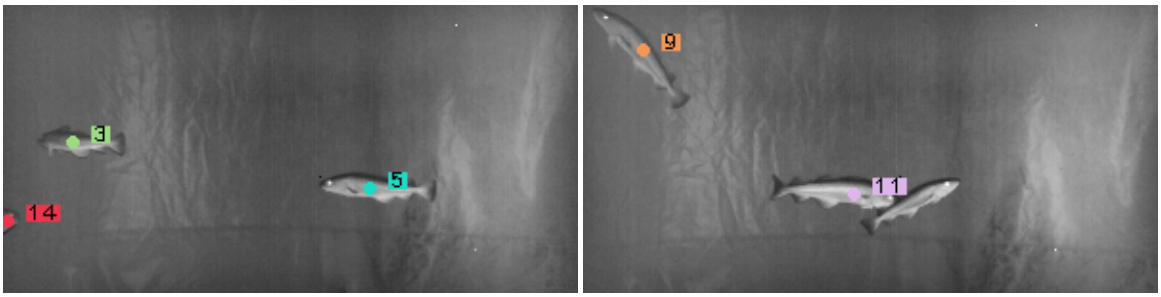


Figure 4.3: Example images from the segmentation validation dataset. The hand-segmented binary mask is marked in red.

The dataset used for testing the tracking methods is a set of 501 subsequent frames. In every frame, the approximate midpoint of each fish was marked and labelled with a unique ID by hand. In merge situations, as shown in Figure 4.4 (b), the merged fish are marked with a single ID. This dataset was used to validate both the tracklet generator and the tracklet linker. When used to validate the tracklet generator, fish do not regain their original ID after a merge as this is the desired output from the tracklet generator. When used to validate the tracklet linker, the ground truth IDs are changed so that the fish do regain their original ID.



(a) Individual fish *(b) Two merged fish with a single ID*

Figure 4.4: Example images from the tracking validation set. Note that the observation with ID 11 in (b) is an observation containing two fish detected as one as they swim close to each other.

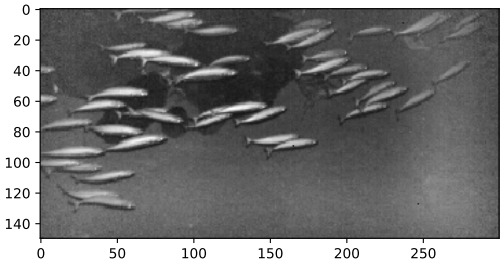
A known issue with the tracking validation set is that the hand annotated dataset will almost certainly mark a merge at a slightly earlier or later frame than the algorithm will, causing ID swaps to be recorded where they should not. This needs to be taken into account when testing, as it may negatively affect the tracking validation results.

4.2 Fish Schools

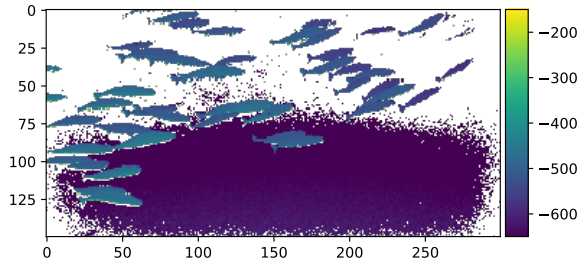
This second dataset is more challenging than the first. It consists of 725 frames in total, filmed in an aquarium containing fish of several species. The dataset includes several schools of fish swimming closely together, as well as several instances of larger fish occluding multiple smaller fish. In addition, large parts of the depth map have None-entries. Similarly to the 3 Cod dataset, the edges of the depth map contain mostly None-entries. Therefore, every frame in this dataset was also cropped from the original size of 480×256 to 300×150 pixels. However, in contrast to the 3 Cod dataset, large parts of the background in the top half of the image are also missing distance entries.

Figure 4.5 shows some examples of the cropped images. Due to the high complexity both in the segmentation and the tracking of the fish in the Fish Schools dataset, the 3 Cod dataset was used to generate most of the experimental results in this thesis. The Fish Schools dataset will be used to illustrate how well the final algorithm performs when applied to

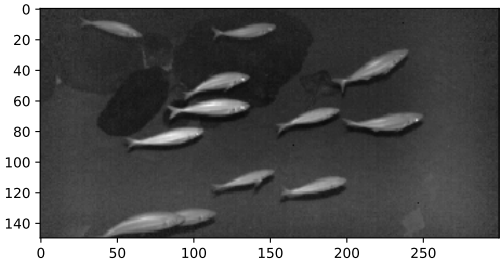
new data, and perhaps how the algorithm needs to be tweaked in order to achieve good results under different circumstances.



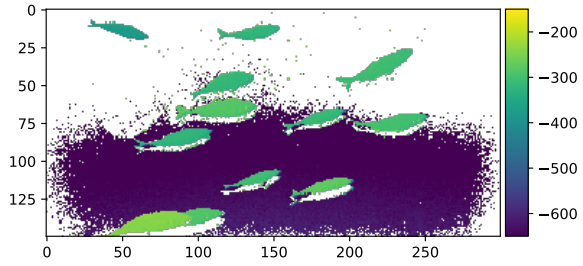
(a) Intensity frame 133



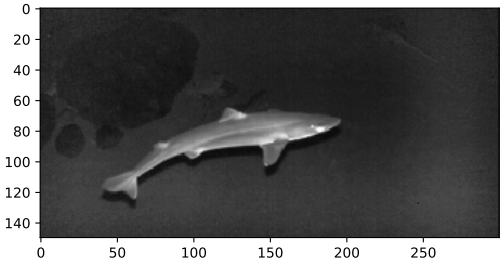
(b) Depth map frame 133



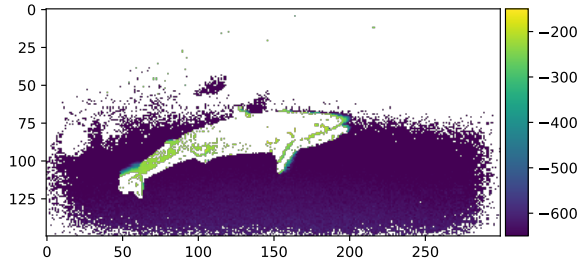
(c) Intensity frame 285



(d) Depth map frame 285



(e) Intensity frame 650



(f) Depth map frame 650

Figure 4.5: Example images from the Fish Schools dataset. The white parts of the depth maps indicate missing (None) entries. Frame 133 likely contains too many fish for the tracker to accurately handle. Note how a large number of fish overlap in the depth map (b); some of these fish are never detected alone. Middle images (c) and (d) illustrate a group of fish more likely to be tracked. Lower images (e) and (f) illustrate how the depth map is missing most information about the large fish in the image; using only the depth map, this fish may be detected as several smaller fish.

Chapter 5

Proposed pipeline

The final algorithm is intended to extract information about individual fish from the input data. The input data consists of a set of N intensity and depth map images. In the first step, a corresponding set of N binary masks is produced, in which probable fish pixels are marked with ones and probable background pixels are marked with zeros. A list of connected components for every frame, computed from the binary mask of that frame, is returned.

In the next step, series of observations with a high probability of belonging to the same fish are linked to form tracklets. The observations are linked by solving the assignment problem for every frame; maximizing the probability of pairing each observation in the current frame with an observation in the previous frame. The probability that two observations are linked is calculated using a cost function consisting of one or more cost features. Cost features describe the similarity between two observations. The choice of a cost feature or cost features is paramount to the success of the tracklet generator.

After successfully generating tracklets, the tracklets are processed to estimate the fish's velocity vector and length. A set of length estimates for each tracklet is produced using every observation in the tracklet. The final length estimate is based on a percentage of the highest length estimates in each tracklet, as the length of the fish is more likely to be underestimated (due to contortions and segmentation errors) than overestimated. The fish's velocity vector is estimated using a Kalman filter. Merges, situations in which multiple fish are detected as a single object as they swim past one another, are not solved by the tracklet generator. Using cost features relying on frames after the merge as well as prior to the merge, such as the velocity vector, makes it possible to solve more difficult observation linking problems.

The tracklet linking step considers every tracklet in the selected time range in order to solve occlusions and missed detection cases correctly. If successful, it will return a set of tracks. Each track includes every observation of a single fish, along with length and speed estimates for every frame it is visible. A final length estimate for each track is also calculated.

The success of the proposed algorithm is dependent on the individual

success of the segmentation and the tracking algorithms (the cost feature selection in particular). Possible solutions to each step are explored in depth in the theory and result sections.

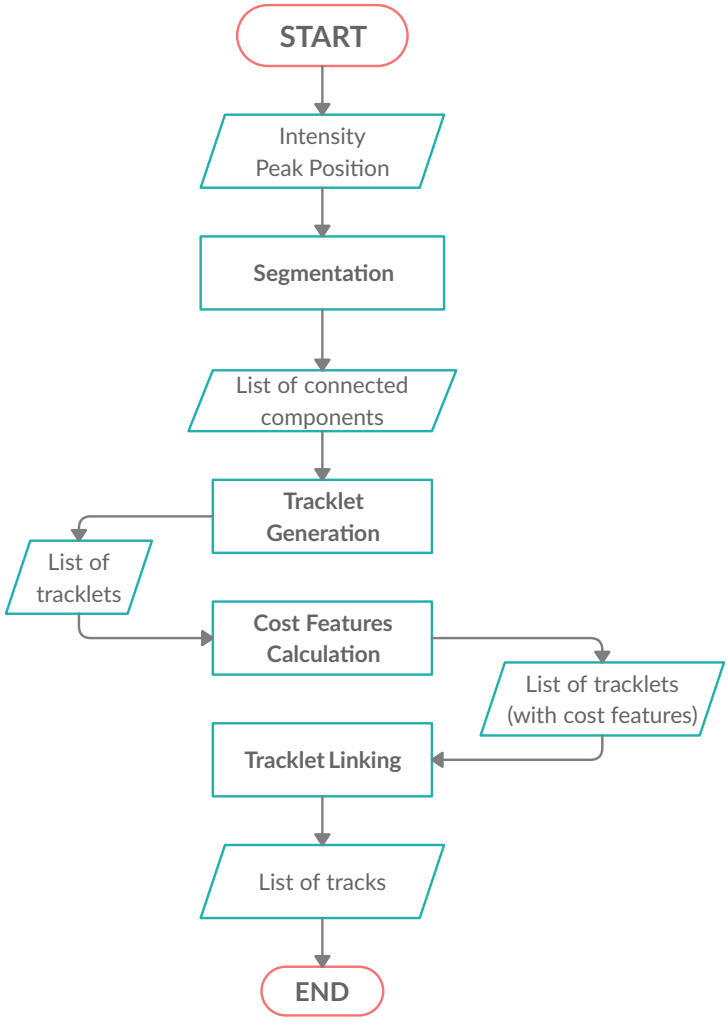


Figure 5.1: Proposed pipeline. After segmentation of every frame of the initial input data, connected components are linked, frame-by-frame, to form short tracklets. The linking is done by solving the assignment problem for every frame with a cost function that estimates the likelihood of two observations belonging to the same fish. The cost function is based on one or more cost features. After the tracklets have been generated, further cost features that cannot be estimated from a single frame are extracted. With a new cost function, tracklets are linked to create tracks. Each track should contain every observation of a single fish. The length estimation algorithm calculates the length for every observation in a track, before the final length is estimated to be a high percentile of the track lengths.

Chapter 6

Segmentation

Image segmentation divides an input image into multiple regions, where each region has a certain characteristic. In this thesis, each frame will be segmented into two regions: the foreground (fish) and the background (everything else). The output of the segmentation process is a binary mask where ones denote pixels belonging to a fish and zeros denote the background.

This chapter contains four main sections. In the first section, I will discuss global and local segmentation techniques based on thresholding. Segmentation based on background subtraction is discussed in Section 6.2. Section 6.3 concerns edge detection, as there are often sharp edges between two regions in an image. Finally, Section 6.4 considers noise removal prior to and after segmentation. As most of these methods can be applied to both the intensity image and the depth map, "intensity" in this section may refer to the value of pixels in the intensity image or the value of pixels in the depth map.

Parts of this chapter are based on my Master Essay "A discussion and comparison of segmentation methods in images," June 2019.

6.1 Thresholding

Thresholding is a way to do a binary partition of an image; useful for separating foreground objects from the background. A threshold T is selected, and any pixel (x, y) with an intensity value I larger than the threshold is given the value one. Pixels where $I(x, y) \leq T$ are given the value zero. Selecting a good threshold is essential for this operation to return meaningful results.

6.1.1 Global thresholding

Otsu's method

In many cases, foreground objects have an intensity distribution that differs from the background, and a carefully selected threshold T might be able to separate the foreground from the background. One of the most popular

algorithms for calculating T is Otsu's method. [59] The method uses class variance as a metric to measure the "goodness of a threshold". [45]

Given an image with G gray levels and histogram probabilities $p(i)$ for each gray level $i = 1, ..., G$, Otsu's method aims to maximize the separability in graylevels of the resulting classes. After applying a threshold T, the probabilities for the two classes C_1 and C_2 are as follows:

$$P_1(T) = \sum_{i=1}^{T} p(i) \tag{6.1a}$$

$$P_2(T) = 1 - P_1(T) \tag{6.1b}$$

The mean values are:

$$\mu = \frac{\sum_{i=1}^{G} ip(i)}{\sum_{i=1}^{G} p(i)} \qquad \mu(t) = \sum_{i=1}^{T} ip(i) \tag{6.2a}$$

$$\mu_1(T) = \frac{\sum_{i=1}^{T} ip(i)}{P_1(T)} \qquad \mu_2(T) = \frac{\sum_{i=T+1}^{G} ip(i)}{P_2(T)} \tag{6.2b}$$

Note that

$$P_1(T)\mu_1(T) + P_2\mu_2(T) = \mu. \tag{6.3}$$

The between-class variance for 2 classes is defined as

$$\begin{aligned} \sigma_b^2 &= P_1(T)(\mu_1 - \mu)^2 + P_2(T)(\mu_2 - \mu)^2 \\ &= P_1(T)P_2(T)(\mu_1 - \mu_2) \end{aligned} \tag{6.4}$$

Using equations 6.1 and 6.3, equation 6.4 can be rewritten as a function of a threshold T.

$$\sigma_B^2(T) = \frac{(\mu_T P_1(T) - \mu_(T))^2}{P_1(T)(1 - P_1(T)} \tag{6.5}$$

To find the value $T^\star$ that maximizes equation (6.5), one may simply evaluate it for all G gray levels for which $0 < P_1(T) < 1$

Through simulations with varying noise, object size and mean difference, Lee and Park [31] showed that Otsu's method produces good results if the histogram have a bimodal distribution with a clear, deep valley between the peaks. In cases with excessive noise, or a smaller difference between the mean intensity values of the object and the background combined with large within-class variances, there are no clear peaks and Otsu's method alone will not produce a good result.

The total error for Otsu's method rapidly increases as $log_{10}(P_1/P_2)$ approaches 1, and Otu's method should only be used when $0.1 < \frac{P_1}{P_2} < 10$. [1] If the object area is small compared to the background, Otsu's method alone is not a good fit.

Kittler-Illingworth's method

In 1985, Kittler and Illingworth proposed a different criterion for an optimal threshold. [27] Assuming the gray level histogram results from an unknown mixture f of two Gaussian distributions, Kittler and Illingworth suggest the threshold T that minimize the Kullback-Leibler divergence J from the observed histogram $p(i)$ to the unknown distribution mixture f. The Kullback-Leibler divergence measures the difference between two probability distributions and is defined: [72]

$$J = \sum_{i=1}^{L} p(i) ln \left(\frac{p(i)}{f(i)} \right) \tag{6.6}$$

for $i = 1, ..., G$ gray levels. We work under the assumption that the observed histogram is the result of a mixture of two Gaussian distributions h_1 and h_2 with means μ_1 and μ_2 and variances σ_1^2 and σ_2^2, and that the probabilities for the two distributions are P_1 and P_2 respectively.

$$\begin{gathered} f(i) = P_1 h_1(i) + P_2 h_2(i) = \\ \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(T-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(T-\mu_2)^2}{2\sigma_2^2}} \end{gathered} \tag{6.7}$$

The criterion J (equation 6.6) can be rewritten:

$$J = \sum_{i=1}^{L} p(i) ln(p(i)) - \sum_{i=1}^{L} P(i) ln(f(i)) \tag{6.8}$$

Realising that the first term does not depend on the unknown parameters in f, we only need to minimize the second term [19, p 23-28] in order to find a minimal J.

$$H = -\sum_{i=1}^{L} p(i) ln(f(i)) \tag{6.9}$$

We assume that the two unknown distributions that make up f are well separated. In this case, $f(i)$ can be written

$$f(i) \approx \begin{cases} \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(T-\mu_1)^2}{2\sigma_1^2}}, & i \leq T \\ \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(T-\mu_2)^2}{2\sigma_2^2}}, & i > T \end{cases} \tag{6.10}$$

for some threshold T separating the two distributions. [19, p 23-28]. Inserting equation 6.10 into equation 6.9 and simplifying, we obtain the following optimization criterion: [42]

$$\begin{aligned} J(T) = 1 &+ 2[P_1(T) ln\sigma_1(T) + P_2(T) ln\sigma_2(T)] \\ &-2[P_1(T) lnP_1(T) + P_2(T) lnP_2(T)] \end{aligned} \tag{6.11}$$

If T is indeed the threshold separating the unknown distributions, the estimated means and variances from the histogram $p(i), i = 1, ..., G$ will be close to the true means and variances $\mu_1, \mu_2, \sigma_1, \sigma_2$ that define f. [19, p 23-28]. $J(T)$ can be evaluated for each threshold T using equation (6.1) to estimate parameters for $P_1(T)$ and $P_2(T)$, equation (6.2b) to estimate $\mu_1(T)$ and $\mu_2(T)$ and the following equation to estimate $\sigma_1(T)$ and $\sigma_2(T)$:

$$\begin{aligned} \sigma_1^2(T) &= \sum_{i=1}^{T} \frac{(i - \mu_1(T))^2 p(i)}{P_1(T)} \\ \sigma_2^2(T) &= \sum_{i=T+1}^{G} \frac{(i - \mu_2(T))^2 p(i)}{P_2(T)} \end{aligned} \tag{6.12}$$

This criterion can be evaluated for every threshold T, or be found through an iterative search. In the latter case, an unfortunate starting value may give a meaningless threshold. Using a start threshold obtained by Otsu's method ensures convergence towards a minimum error threshold. In this thesis, an exhaustive search was used to find the optimal threshold.

Previous experimental results show that Kittler-Illingworth's method performs better than Otsu's method on images with small objects compared to the background. [37] In 2004, Sankur and Sezgin evaluated the performance of 40 thresholding methods, including Otsu's method and Kittler-Illingworth's method, based on five different performance criteria. They ranked Kittler-Illingworth's method first. [59] Albregtsen compares eight thresholding methods, including Otsu's method and Kittler-Illingworth's method, on normalized binormal histograms with varying probability ratio and distance between the peaks. [1] Kittler-Illingworth's method has a stable error rate even for larger probability ratios, while the error rate for Otsu's method increases rapidly as the probability ratio grows. Albregtsen notes that Kittler-Illingworth is always close to the ideal error rate. [1]

6.1.2 Adaptive thresholding

In some cases, for instance when an image has some intensity gradient, a global threshold is not optimal for separating foreground and background elements. In these cases, adaptive thresholding methods might prove to be a better solution. When adaptive thresholding methods are used, the value of T is calculated for every pixel and changes depending on the intensity values in a local neighbourhood.

Adaptive mean and Gaussian thresholding

A common threshold chosen is the mean value for a local neighbourhood of size $w \times w$ minus a constant C. Increasing the constant C results in larger parts of the image being segmented. This value may be set to 0 or even negative.

The Gaussian adaptive thresholding algorithm utilizes a weighted sum of the local neighborhood instead of the mean, ensuring that close neighbours have a higher influence on the threshold than more distant neighbours. The threshold for each pixel (x, y) is calculated by cross-correlating each neighbourhood window of size $w \times w$ with a Gaussian kernel $g(\sigma)$ of the same size:

$$T(x, y) = g(\sigma) \star I(x, y) - C \tag{6.13}$$

σ is calculated from the size w of the local neighborhood: $\sigma = 0.3((w-1) * 0.5 - 1) + 0.8$. [44]

Niblack's method

Developed in 1986, Niblack's local threshold method is based on the mean and standard deviation in a local window. The threshold T_w in a local window w is calculated as follows:

$$T_w = m_w + k \times \sigma_w \tag{6.14}$$

m_w and σ_w represents the mean and standard deviation in the window. The constant k and the window size w are predetermined; Niblack suggested $k = -0.2$. [4] This method tends to correctly identify the foreground, but returns large amounts of noise in empty areas of the image. [25]

Sauvola's method

Sauvola's method from 1997 attempts to fix the noise problems in Niblack's method. The approach is similar to Niblack's method, but it includes a constant gray-level range value R, set to 128 for 8-bit images by the authors. [60] The thresholding formula for each window is as follows:

$$T_w = m_w \left(1 - k \left(1 - \frac{\sigma_w}{R}\right)\right) \tag{6.15}$$

where k is a constant in the range $[0.2, 0.5]$. [4] While this method largely solves the noise problem in Niblack's method, it struggles when the contrast between the foreground and the background is relatively small. [4]

Bataineh's method

This method from 2011 was an attempt to address the weaknesses of the previous methods, including noise, low contrast and the issue with manually choosing parameters. Bataineh's method uses the mean and standard deviation of both the local window and the entire image in the threshold calculation: [4]

$$T_w = m_w - \frac{m_w^2 \times \sigma_w}{(m_g + \sigma_w) \times (\sigma_a + \sigma_w)} \tag{6.16}$$

m_w and σ_w are the mean and standard deviation of the local window. m_g represents the mean of the entire image. σ_a is an adaptive standard deviation calculated using the overall image minimum and maximum standard deviation values:

$$\sigma_a = \frac{\sigma_w - \sigma_{min}}{\sigma_{max} - \sigma_{min}} \tag{6.17}$$

6.2 Background subtraction

In some scenarios, for instance when a camera is affixed to a trawl pointing down towards the sea bed, the depth map will have a fairly constant, homogeneous background with the fish being closer to the camera. It may be possible to segment the fish by subtracting a model of the background from each frame. The background is not always a flat, planar surface, therefore a simple threshold is not sufficient. Instead, the background is modelled as a function $f(x, y)$ of polynomial degree d:

$$\hat{z} = f(x, y) = \sum_{j=0}^{d} \sum_{i=0}^{j} A_{ij} x^i y^{d-i} \tag{6.18}$$

Sampling n random points in the depth map with coordinates (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) and distances $\mathbf{z_r} = [z_1, z_2, ..., z_n]^T$, a linear system is constructed. Equation 6.19 shows a linear system of degree $d = 2$. This method easily extends to higher dimensions.

$$\underbrace{\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & x_2 y_2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & x_n y_n & x_n^2 & y_n^2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}}_{\hat{\mathbf{z}}} \tag{6.19}$$

The system of linear equations is solved by using the ordinary least squares method (OLS) to find the **b** that minimizes the cost function:

$$C = \frac{1}{n} \sum_{k=1}^{n} (z_k - \hat{z_k})^2, \tag{6.20}$$

where $z_k, k = 1, ..., n$ are the measured distances at the randomly chosen coordinates and $\hat{z_k} = f(x_k, y_k), k = 1, ..., n$ are the distances estimated with equation 6.18. [22]

After solving the linear system, the segmentation is completed by computing the absolute difference $D(x, y)$ between the depth map background estimate $\hat{z}$ and the current depth map frame z. A mask $M(x, y)$ is created:

$$M(x, y) = \begin{cases} 1, & \text{if } D(x, y) > T \\ 0, & \text{otherwise} \end{cases} \tag{6.21}$$

While there are more robust methods available for solving such systems of linear equations, OLS seems to be sufficient to model the background

in this case. If too many foreground object (fish) points are included in the background estimation, the mask will likely include a large number of false positives. This may be avoided by iteratively calculating the background, selecting new random points for the fitting function every time, and keeping only areas of the image consistently marked as foreground. Careful selection of the segmentation threshold T, the polynomial degree d and the number of random points used for the plane fitting may also help.

6.3 Edge detection

If the background is fairly smooth and homogeneous, edge detection can be used to capture the contour of the fish. The goal of edge detection algorithms is generally to mark connected lines in the image at locations where there is a sudden change in pixel intensity values, for instance the sudden change in the depth map from a background pixel to a fish pixel.

In this thesis, edge detection is used in conjunction with a segmentation method to improve the segmentation near the edges of the fish.

6.3.1 Canny's edge detector

One of the most popular edge detection algorithms is Canny's edge detector. The edge detector aims to consistently detect only true edges, place the edge marker point in the center of the true edge and eliminate multiple responses to a single edge point. [7] These criteria were expressed mathematically and solutions approximated using numerical optimization. Modelling a step edge in 1-D with added white Gaussian noise, Canny found that the first derivative of a Gaussian was a sufficient approximation of the optimal step edge detector. [17]

In 2D, one needs to apply the 1D edge detector in all directions to find the edge normal, the direction for which the 1D results still applies. This can be accomplished by first smoothing the image and then computing the gradient magnitude and direction. [17, p. 729-732]

$$I_s(x,y) = g(x,y) \star I(x,y) \tag{6.22}$$

The image $I(x,y)$ is smoothed by convolving with a Gaussian function (equation 6.23) to create $I_s(x,y)$.

$$g(x,y) = e^{\frac{x^2+y^2}{2\sigma^2}} \tag{6.23}$$

Next, the gradient magnitude $||\nabla I_s(x,y)||$ and direction $\theta(x,y)$ are computed. The horizontal and vertical gradients can be found by using Sobel operators which yields g_x and g_y.

$$||\nabla I_s(x,y)|| = \sqrt{g_x(x,y)^2 + g_y(x,y)^2} \tag{6.24}$$

$$\theta(x,y) = tan^{-1}[\frac{g_y(x,y)}{g_x(x,y)]} \tag{6.25}$$

Non-maximum suppression is used to thin the wide ridges around local maxima, resulting in the non-maximum suppression image I_n. Let K denote the value of $||\nabla I_s||$ at a detected edge point (x, y). Search a predefined local neighbourhood around (x, y). If K is less than the value at a neighbouring point in the direction of the edge, the point (x, y) is suppressed ($I_n(x, y) = 0$). Otherwise, $I_n(x, y) = K$. [17].

The gradient magnitude image is thresholded to define the edges. In order to suppress false positives, Canny's edge detector uses two thresholds T_H and T_L, so-called hysteresis thresholding, to create two thresholded images.

$$I_{nL}(x, y) = I_n(x, y) \leq T_H \tag{6.26}$$

$$I_{nH}(x, y) = I_n(x, y) \leq T_L \tag{6.27}$$

Note that $T_H > T_L$, meaning I_{nL} will contain all non-zero pixels in I_{nH} and some additional. The non-zero pixels in I_{nL} not in I_{nH} are denoted weak edges, while all non-zero pixels in I_{nH} are denoted strong edges. All strong edge points are initially marked as true edge points. A pixel point in a weak edge is suppressed if it is not connected to a strong edge point using 8-connectivity. If a pixel point in a weak edge is connected to a strong edge, the point is marked as a true edge point.

Utilizing hysteresis thresholding will form longer, connected edges and preserve true edge points that would otherwise have been suppressed without picking up on too much noise.

6.4 Noise removal

Underwater images are especially subjected to noise, more so that images captured in air. As light travels through water, it is exponentially attenuated, which, depending on the turbidity of the water, can severely limit the visibility distance. Forwards and backwards scattering blurs and lowers the contrast of the intensity image, [61] and may introduce false peaks in the depth map as discussed in Chapter 3. These factors affect the segmentation of both the intensity image and the depth map.

Bilateral filtering can be used to smooth the depth map and still keep the important sharp transition from the background distances to a fish intact. Morphological operations can be used post-segmentation to remove small noise particles.

6.4.1 Bilateral filtering

In general, smoothing of an image is done using a lowpass filter; a filter that reduces the difference between pixel values by replacing the intensity of each pixel with a (sometimes weighted) average of nearby pixels. [47] Let $I(\mathbf{p})$ represent the original intensity value and $H(\mathbf{p})$ be the output of the lowpass filter at pixel position $\mathbf{p} = (x, y)$. $H(\mathbf{p})$ is calculated:

$$H(\mathbf{p}) = \sum_{\mathbf{q} \in S} g(\|\mathbf{p} - \mathbf{q}\|) I(\mathbf{q}) \tag{6.28}$$

Here, $\mathbf{q}$ are pixel positions in the window S and g is a weighting function. A popular choice for g is a normalized Gaussian function:

$$g(\mathbf{x})_\sigma = \frac{1}{2\pi\sigma^2} exp(-\frac{\mathbf{x}^2}{2\sigma^2}) \tag{6.29}$$

where a large σ gives more weight to distant locations and thus results in more blurring. [68]

The lowpass filter will smooth the edges along with the rest of the texture in the image. However, a bilateral filter preserves the edges while smoothing non-edges and is thus useful for removing texture and noise in the image while keeping important edge information. The filter is based on a low-pass filter, but with an additional normalization term. The normalization term ensures that not only spatial location but also pixel intensity ranges determine whether two pixels are close to each other. [68]

$$H(\mathbf{p}) = W_p^{-1}(\mathbf{p}) \sum_{\mathbf{q} \in S} g_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) g_{\sigma_r}(I(\mathbf{p}) - I(\mathbf{q})) I(\mathbf{q}) \tag{6.30}$$

S represents all possible pixel positions in the image, and g is a weighting function. [47] $W_\mathbf{p}$ is defined:

$$W_\mathbf{p} = \sum_{\mathbf{q} \in S} g_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) g_{\sigma_r}(I_\mathbf{p} - I_\mathbf{q}) \tag{6.31}$$

The spatial variance σ_s influence the amount of blurring, while the intensity range variance σ_r controls which features will be smoothed. [68] For a large σ_s, the bilateral filter resembles the low-pass filter. Increasing σ_r allows intensity values with larger differences between them to be smoothed. [47] In neighbourhoods with more uniform intensities, the normalization will be close to one and the output is similar to a standard low-pass filter.

6.4.2 Morphological operations

Morphological operations are very useful in processing and extracting information about geometrical structures, such as objects in a binary image. We will mainly use them to remove small noise elements from binary images. The operations are based on set theory, where the sets are collections of 2D coordinates. Two types of sets are typically used: objects and structuring elements. In this case, objects are defined as sets of foreground pixels (pixels with value equal to one in a binary image) while structuring elements can contain both foreground and background pixels. [17, p. 636]

Let S denote the set of all image coordinates in an image. We define two morphological operations based on vector addition of set elements of an object A and a structuring element B. Note that A and B are subsets of S. [20]

$$A \oplus B = \{c \in S | c = a + b \text{ for some } a \in A \text{ and } b \in B\} \tag{6.32}$$

The dilation of set A using structuring element B, denoted $A \bigoplus B$ is "the set of all possible vector sums of pairs of elements, one coming from A and one coming from B." [20]. When applying dilation to an binary image, the objects in the image will "grow" according to the shape and size of the structuring element. Dilation can, for instance, help connect parts of an object if the object was only partially segmented.

$$A \ominus B = \{c \in S | c + b \in A \text{ for every } b \in B\} \tag{6.33}$$

The erosion of a set A using the structuring element B is the set of all points c such that "B, translated by [c], is contained in A". [17, p. 639] It "shrinks" A according the shape and size of B and is useful in removing small noise elements/false positives from binary images as it will erase objects smaller than the structuring element.

Erosion and dilation can be combined to form further morphological operations; two important ones being opening and closing. Both will generally smooth the contour of an object. Opening removes sharp peaks and eliminates small islands, while closing bridges narrow gaps and eliminates small holes. [20]

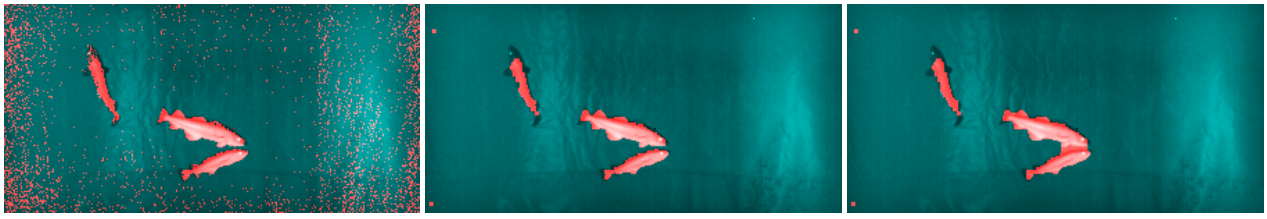
The opening of a set A using a structuring element B is defined as erosion of A by B followed by the dilation of the result by B:

$$A \circ B = (A \ominus B) \oplus B \tag{6.34}$$

The closing of a set A using structuring element B is defined as the dilation of A by B followed by the erosion of the result by B: [17, p. 645]

$$A \bullet B = (A \oplus B) \ominus B \tag{6.35}$$

The example images in Figure 10.20 illustrate how opening and closing can remove noise from the segmentation.



(a) Original segmentation output *(b) After morphological opening* *(c) After morphological closing*

Figure 6.1: Morphological opening removes most of the small noise particles. Morphological closing eliminates the gap between two of the fish.

Chapter 7

Tracking

Tracking the fish is not an easy task. The three codfish in the 3 Cod dataset are relatively similar in size and appearance. The intensity and the shape of the fish, two features commonly used in tracking, continuously change as the fish change direction. The direction and speed of the free-swimming fish change frequently as well, making tracking based on position estimates with Kalman filters difficult. This thesis uses a tracking algorithm based on a minimum-cost flow network as it is easily customisable and includes occlusion handling.

A cost-flow network is an intuitive way to represent a set of tracks. Let every observation be a node in the graph. Directed arcs, or edges, link observations together, creating a set of possible tracks. An observation may also be linked to a source or sink node, representing the fish entering and exiting the frame respectively. The weight of the edges represents the likelihood of linking two observations. The graph defines a maximum a posteriori (MAP) problem. Figure 7.1 illustrates what one such graph along with a possible solution may look like. Weak cost features in the cost function calculating the weight of each edge is the main cause of failure for this algorithm.

In this thesis, observations in neighbouring frames are initially linked to form tracklets; sets of observations belonging to the same fish. Features such as the fish's velocity vector and length can be estimated from the tracklets. Tracklets are linked to form the final tracks, using features calculated based on multiple observations. Occlusions and missed detections can be resolved at this stage using information prior to the "loss" of the fish and after the fish has been detected again.

7.1 Tracklet generation

Consider a set of n observations $O^t = o_1^t, o_2^t, ..., o_n^t$ detected in frame t. Our goal is to find the most likely set of edges that connects each observation in frame t to an observation in a set of m observations in the previous frame $t-1$.

The most likely edges connecting the two sets of observations can be found by solving the assignment problem. Figure 7.2 illustrates a graph

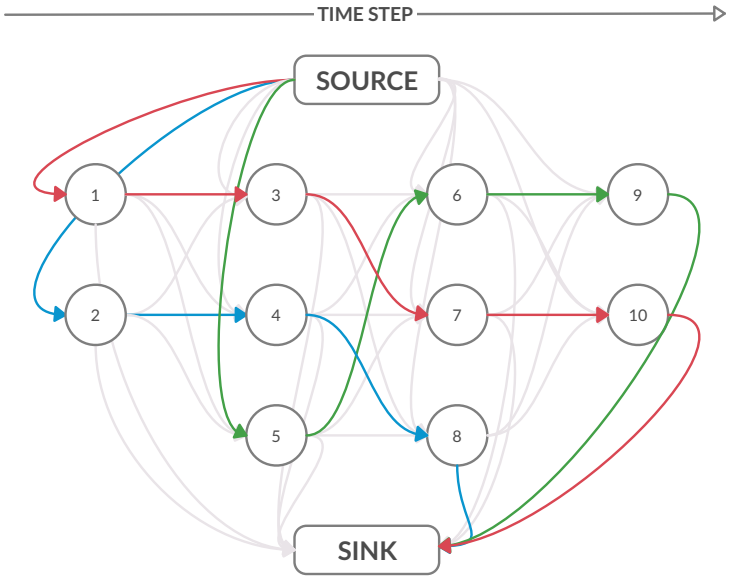


Figure 7.1: An example of a cost-flow network for tracking. Each node represents an observation. There is a possible flow from the source node to every observation, as well as a possible flow from every observation to the sink. For every node in time step t, there is a possible flow from every node in time step $t-1$. An example solution with three tracks is marked in color. There is no flow along the grey arcs. Note that this network does not allow multiple incoming or outgoing flows for the observation nodes, meaning the network must be adjusted in order to correctly solve merge situations.

$G(O^{t-1}, O^t, E)$ with two sets of nodes (observations) and a set of edges E. Each node $o_i^{t-1} \in O^{t-1}$ is connected to all nodes $o_j^t \in O^t$ by an edge $e_{ij} \in E$ with a weight c_{ij}. The weight represents how likely it is that observation $o_i^{t-1} \in O^{t-1}$ and $o_j^t \in O^t$ belong to the same fish. The assignment problem asks for the bipartite graph $G'(O^{t-1}, O^t, E')$, where each node in O^{t-1} is connected to one node in O^t by an edge in $E' \subset E$ such that the sum of the weights is minimized. [51]

The assignment problem can be solved using the Hungarian method in time $O(nk + n^2 log(n))$ if $|O^{t-1}| = |O^t| = n$ and $|E| = k$. [51] However, the number of observations in two neighbouring frames will frequently be different, for instance due to occlusions or on occasions where fish have entered or exited the frame. That is, the graph is unbalanced. The simplest way to transform an unbalanced graph into a balanced one is to add enough new nodes with zero-cost edges connecting them to all the nodes in the larger part, until $|O^{t-1}| = |O^t|$. This is not efficient for larger optimization problems, but in this case the number of observations per frame will remain small. [51] As illustrated in Figure 7.2, an observation

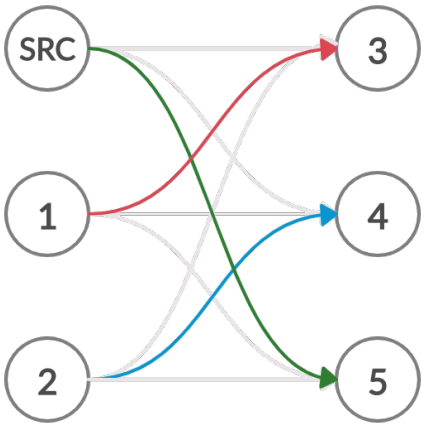


Figure 7.2: A section of the graph in Figure 7.1. At time step $t-1$, there are two observations ($m = 2$) represented by node 1 and 2. At time step t, a new fish has appeared in the frame ($n = 3$). Therefore, a new node SRC is added to the left side so that $m = n$ and the solution marked in colors may be found using the Hungarian method.

being connected to a non-observation node v represents a connection to the source if v is added to O^{t-1} or the sink if v is added to O^t. An observation being connected to the source or the sink represents the fish entering or exiting the frame respectively.

7.1.1 Cost matrix

Let c_{ij} represent the cost of linking o_i^{t-1} and o_j^t. The cost is calculated by taking the negative logarithm of a linking probability $P_{link}(o_i^{t-1}, o_j^t)$, meaning that a high cost indicates a low likelihood that two observations belong to the same fish. The linking likelihood is calculated as the product of a set of cost features (discussed further in Section 7.3), normalized with a zero-mean Gaussian function.

$$c_{ij} = -logP_{link}(o_i^{t-1}, o_j^t) \tag{7.1}$$

For every frame t, the cost of linking every observation $o_j \in O^t, j = 1, ..., n$ with every observation $o_i^{t-1} \in O^{t-1}, i = 1, ..., m$ in the previous frame is calculated and represented in an $n \times m$ cost matrix C:

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ \vdots & \vdots & & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{bmatrix} \tag{7.2}$$

If $m > n$, extra zero-rows are added until the matrix is square (representing connections to the source). If $m < n$, extra zero-columns are added until the matrix is square (representing connections to the sink).

7.1.2 The Hungarian method

After the cost matrix has been calculated, the Hungarian method[1] may be used in order to find the bipartite graph which minimizes the total cost of the edges. This method was developed in 1955 by Harold Kuhn [28] and can be described in six steps for a cost matrix of size $n \times n$: [49]

1. Find the minimum value m_i in each row and subtract m_i from every entry in that row.
2. For each zero Z in the resulting matrix, check if there is a starred zero in its row or column. If not, star Z.
3. Cover the columns containing a starred zero. Check for optimality: if n columns are covered, the starred zeros indicate the optimal solution. If the optimality test fails, go to step 4.
4. Prime an uncovered zero. If its row contains a starred zero, cover the row and uncover the column containing the starred zero. Repeat until there are no uncovered zeros. Go to step 6. If the row contains no starred zeros, go to step 5.
5. Start a process to construct a series $Z_0, Z_1, Z_2...$ of alternating primed and starred zeros. Let the uncovered zero found in step 4 be Z_0. Z_1 denotes the starred zero found in the column of Z_0 (if any). Let Z_2 be the primed zero in the row of Z_1. Continue the series until a primed zero is found that has no starred zero in its column. Unstar each starred zero and star each primed zero of the series. Remove all primes, and uncover every line in the matrix. Return to step 3.
6. Find the minimum uncovered value. Add it to every covered rows, and subtract it from every uncovered column. Return to step 4.

Steps 3-4 will repeat until an optimal solution is found. Note that the process of constructing the series of primed and starred zeros will increase the number of assignments by one. [28] The number of assignments are bounded above by n, since each label can be connected to at most one observation.

7.2 Tracklet linking

Using the approach described previously, single observations have been connected to form a set of tracklets. No attempt has been made so far to solve merges and splits. It should also be noted that the method above will restart a tracklet if even a single observation is missed by the detector. I hope to solve these issues by solving an extended minimum-cost flow network graph problem, similar to the one described above with some modifications.

[1] Kuhn referred to his method as "the Hungarian method" because it was based on the work of two Hungarian mathematicians: Dénes Kőnig and Jenő Egerváry.

Let $\mathcal{T} = \{T_j\}$ be a set of tracklets created by linking observations frame-by-frame using the Hungarian method. Each tracklet $T_j = (O_j, A_j, t_j)$ contains a list of successive detections with high probability of association O_j, a number of descriptors used in the calculation of cost features (for instance length in centimetre) A_j and the time step the first observation appeared t_j. The tracklet generator algorithm should be designed to somewhat aggressively end tracklets if a highly probable link to an observation cannot be found. Therefore, we assume that no ID swaps occur in the tracklets; meaning that each tracklet either contains observations of a single fish or observations of multiple fish in an occlusion situation.

A track is defined as an ordered list of tracklets: $S_k = \{T_{k_1}, T_{k_2}, ..., T_{k_l}\}$. A candidate solution is a set of tracks: $S = \{S_i\}$ and the goal is to maximize the probability of S given a set of tracklets $\mathcal{T}$. It is customary to add the restriction that no tracks may share a tracklet to reduce the solution space ($S_i \cap S_j = \emptyset, \forall i \neq j$). [21] [74]

7.2.1 Extended cost matrix for tracklet linking

The tracklet linking algorithm uses an extended version of the cost matrix described earlier. Tracklet likelihoods, termination likelihoods and initialization likelihoods are calculated in addition to the likelihood of linking tracklets.

The initialization and termination likelihoods are based on the reasonable assumption that new objects only appear near the image borders.

$$P_{init}(T_a) = \mathcal{N}(d_b(O_{a,1}), \sigma_b) \tag{7.3a}$$
$$P_{term}(T_a) = \mathcal{N}(d_b(O_{a,-1}), \sigma_b) \tag{7.3b}$$

d_b represents an observation's distance to the image border. $O_{a,1}$ and $O_{a,-1}$ represent the first and last observation in T_a respectively. σ_b is a fixed parameter based on the expected distance from the image edge detection or loss of detection will occur.

The likelihood of a tracklet T_a being accepted P^+ or discarded P^- is based on the miss rate of the detector β and the length of the tracklet $|T_a|$: [21]

$$P^+(T_a) = (1 - \beta)^{|T_a|} \tag{7.4a}$$
$$P^-(T_a) = \beta^{|T_a|} \tag{7.4b}$$

In this thesis $\beta = 10^{-4}$, calculated from the rate of false negatives in the test set.

Finally, the likelihood of a link from a tracklet T_a to a tracklet T_b is defined, as in the previous section, as the product of n_s carefully selected normalized similarity features $P_f^i, i = 1, ..., n_s$:

$$P_{link}(T_a, T_b) = \prod_{i=1}^{n_s} P_f^i(T_b \mid T_a) \tag{7.5}$$

Let the final observation in T_a occur in frame t_a, and the first observation of T_b occur in frame t_b. The tracklets can be linked if and only if $t_a < t_b$, otherwise $P_{link}(T_a, T_b) = 0$.

In order to accommodate the initialization and termination blocks, the cost matrix for n tracklets is a $2n \times 2n$ square matrix with the following structure: [21]

$$C = \begin{bmatrix} C_{link} & C_{term} \\ C_{init} & \mathbf{0}_{n \times n} \end{bmatrix}$$

The $C_{link} = [c_{ij}]$ block handles tracklet-to-tracklet linking and tracklet likelihoods: [21]

$$c_{ij} = \begin{cases} -log P_i^-, & \text{if } i = j \\ -log\left[\sqrt{P_i^+} P_{link}(T_i, T_j) \sqrt{P_j^+}\right], & \text{otherwise} \end{cases} \tag{7.6}$$

C_{term} and C_{init} are diagonal matrices with tracklet termination and initialization costs along the diagonals and infinity everywhere else. [21]

$$C_{init} = \text{DIAG}_\infty(-log\left[P_{init,k}\sqrt{P_k^+}\right]) \tag{7.7}$$

$$C_{term} = \text{DIAG}_\infty(-log\left[P_{term,k}\sqrt{P_k^+}\right]) \tag{7.8}$$

Using the Hungarian method with input C, an optimal solution may be found. Tracklets being linked to themselves in the optimal solution indicates a false detection.

7.2.2 Resolving merges and splits

The approach described above needs to be further augmented in order to handle merge situations. After finding a solution S, a search is done for possible merge situations. These situations are identified by a tracklet terminating or initializing far from the border and near another tracklet, as illustrated in Figure 7.3.

Consider a pair of tracklets, T_a and T_b, that both end in time step t_f, and a tracklet T_m whose first observation occur in time step $t_m = t_f + 1$. Assume T_a was linked to T_m and T_b was linked to the sink node in the solution S. If the final observation of T_b occurred relatively far away from the border and relatively close to the first observation of T_m, it is likely that T_m is a tracklet consisting of observations of the two fish in T_a and T_b, one occluding the other or the fish swimming so close together that they are not detected as separate fish.

In such cases, an occlusion hypothesis is added to $\mathcal{T}$. [74] T_m is removed from $\mathcal{T}$, replaced by two new tracklets $T_m^a(O_m, A_a, t_m)$ and $T_m^b(O_m, A_b, t_m)$. The new tracklets have the same list of observations and start frame as T_m, but the descriptors of T_a and T_b.

In order to solve situations where two fish are merged as they enter the frame and later split, or merge situations with more than two fish, the

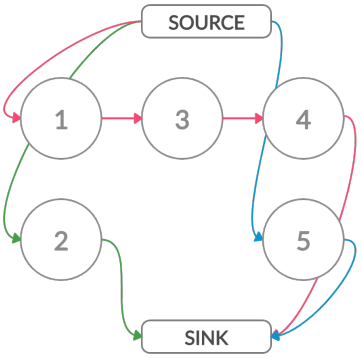


Figure 7.3: An unsolved merge situation. Three tracks are illustrated, marked with different colors. Each node represents a tracklet. Tracklet 3 contains observations of the fish in tracklet 1 and tracklet 2, swimming too close together to be detected as two fish. As only one incoming and one outgoing edge is allowed for each node, only tracklet 1 is connected to tracklet 3, while tracklet 2 is terminated and reinitialized as tracklet 5.

algorithm searches for splits as well as the merges described above. Splits are indicated by pairs of tracklets, T_a and T_b, that both initialize in time step t_s far from the border, near the final detection of a tracklet T_m terminating at time step $t_s - 1$.

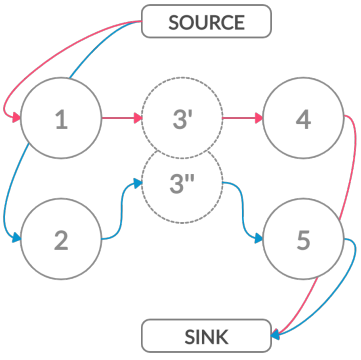


Figure 7.4: Tracklet 3 has been replaced by two copies of itself with descriptors from tracklet 1 and tracklet 2. The cost matrix has been re-calculated, and the new solution is the correct one.

Figure 7.4 illustrates how the merge situation in Figure 7.3 was correctly resolved after an occlusion hypothesis was added. The situation in Figure 7.4 is a fairly simple one, albeit common. The merge may be more complicated if more than two fish merges. Therefore, the final algorithm attempts to solve merges recursively, until no further merge hypothesis is created or a threshold for the number of attempts has been reached.

7.3 Cost features

A good cost function is paramount to the success of the tracking algorithm. Two different cost functions will be developed; one cost function that links observations frame-by-frame to form tracklets, and one cost function that links tracklets in order to form the final tracks.

The cost function is the product of a number of cost features, each cost feature normalized as a zero-mean Gaussian function. The normalized cost features represent the probability that two observations (or tracklets) o_i and o_j belong to the same fish. A good cost feature, prior to the normalization, is expected to return values near zero when o_i and o_j belong to the same fish, and values in a higher range when o_i and o_j belong to different fish.

Two cost features that perform well separately may not always perform well together, due to high correlation between the cost features. The cost features used for frame-by-frame tracking may be different from the tracklet linking cost features, as the latter features are required to remain relatively constant over longer time periods.

7.3.1 Histogram comparison

The intensity distribution of an object in a gray-scale image describes the overall brightness of the object. Due to the high frame rate of the camera, this distribution will likely be relatively consistent across neighbouring frames; thus a metric that describes the similarity between two distributions may be used to link observations frame-by-frame.

Minkowski distance

The Minkowski distance D_{L_p} is a straight-forward, bin-by-bin similarity metric that considers the pair-wise difference between each bin in the two histograms. [56]

$$d_{L_p}(H, G) = \left(\sum_{i}^{N} |h_i - g_i|^p \right)^{\frac{1}{p}} \tag{7.9}$$

where H and G are the two distributions and N is the number of bins. p is typically 1 or 2, equivalent to the Manhattan and Euclidean distance respectively. [56]

The Earth Mover's Distance (Wasserstein Metric)

The Earth Mover's Distance (EMD), also known as the Wasserstein metric, measures the minimum distance between two distributions. Informally, EMD is often described as the least amount of work required to fill a collection of holes in the ground with some piles of dirt. The holes represent one distribution, the piles of dirt represent the other. [56]

Let the intensity histogram $\{h_i\}$ be represented as a signature $\{s_j = (m_j, w_j)\}$. The signature is a set of feature clusters represented by its mean

m_j and the fraction of pixels belonging to that cluster w_j. If $s_j = (m_j, w_j)$ represents the bin h_i in a histogram, w_j corresponds to the number of pixels in that bin, while m_j is the central value in bin i. [56] Computing EMD is based on the transportation problem: a set of suppliers (the piles of dirt) with a certain amount of goods (the amount of dirt in each pile) are required to supply a set of consumers (the collection of holes), each with a limited capacity (depth of the hole).

This can be formalized as a linear program problem. Let $P = (p_1, w_{p1}), (p_2, w_{p2}), ..., (p_n, w_{pn})$ be the first signature consisting of n clusters and $Q = (q_1, w_{q2}), (q_2, w_{q2}), ..., (q_m, w_{qm})$ be the second signature with m clusters. A flow between the two signatures is a matrix $F = [f_{ij}]$, where f_{ij} is the amount of weight at p_i matched at q_j. [10] A flow is feasible if and only if it fulfils the following constraints: [56]

$$f_{ij} \geq 0 \quad i = 1, ..., n, j = 1, ..., m \tag{7.10a}$$

$$\sum_{i=1}^{n} f_{ij} \leq w_{pi} \quad i = 1, ..., n \tag{7.10b}$$

$$\sum_{j=1}^{m} f_{ij} \leq w_{qj} \quad i = j, ..., m \tag{7.10c}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij} = min\left(\sum_{i=1}^{n} w_{pi}, \sum_{i=j}^{m} w_{pj}\right) \tag{7.10d}$$

The first constraint allows movement from P to Q only. The second constraint ensures that one cannot move more "dirt" from a cluster in P than the cluster contains, while the third constraint limits the amount of "dirt" a hole in Q can receive to the depth of the hole. [10] The final constraint forces the total amount of weight to match the lighter distribution; meaning that the maximum possible amount of weight is moved. [56] After solving the transportation problem, one is left with the EMD:

$$EMD(P, Q) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij} d_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}} \tag{7.11}$$

Here, d_{ij} represents the L_1 distance between clusters p_i and q_j. Note that the EMD is normalized by the total flow. [56] For distributions with different total weights, this normalization helps ensure that the smaller signature is not favoured. [56] Unlike the Minkowski distance and other bin-by-bin methods, the distance between bins is considered instead of simply comparing corresponding bins. EMD also allows for partial matches. [56]

7.3.2 Position and velocity

Assuming some distance between the fish, an observation o_i in frame t should in almost every case be matched with the closest observation o_j in frame $t - 1$. Euclidean distance may fail in situations where fish are

swimming close together, but is otherwise likely to be a useful cost feature for frame-by-frame observation linking.

After tracklets have been generated, we can apply a Kalman filter [24] to estimate the velocity vector of the fish from noisy, multidimensional midpoint coordinate measurements. Kalman filters are used to filter out noise from a series of measurement variables by recursively updating an estimate of the state of a linear system. For each time step, the state variable x_t is estimated from the previous state x_{t-1} using equation 7.12. The relationship between the state variable and the measurement z_t is modelled according to equation 7.13. [8]

$$x^t = F \cdot x_{t-1} + B \cdot u_{t-1} + w_{t-1} \tag{7.12}$$

$$z^t = H \cdot x_t + v_t \tag{7.13}$$

x_t and z_t represent the estimated state variable and measurement variable at time t. The state transition matrix is denoted F, the measurement matrix is denoted H. B and u represent the control-input matrix and the control vector respectively. In some systems, external variables such as sudden braking or acceleration are known. In this case, however, these variables are unknown and thus B is set to zero. [66] System noise and measurement noise are assumed to be constant, white and independent of each other, and represented by w_t and v_t, modelled as normal probability distributions: [29]

$$w_{t-1} \sim \mathcal{N}(0, Q) \tag{7.14}$$

$$v_t \sim \mathcal{N}(0, R) \tag{7.15}$$

The algorithm is generally divided into two steps: the prediction step and the update step. In the prediction step of the algorithm, the current state of the system x_t^- and state covariance P_t^- are predicted according to equations 7.16. Note that in this step, only previous measurements are taken into account. [29]

$$x_t^- = F_{t-1} \cdot x_{t-1} + Bu_t \tag{7.16a}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{7.16b}$$

In the update step, the a posteriori state of the system is estimated with equations 7.17:

$$y_t = z_t - H\hat{x}_t^- \tag{7.17a}$$

$$K_t = P_t^- H^T (HP_t^- H^T + R)^{-1} \tag{7.17b}$$

$$\hat{x}_t^+ = \hat{x}_t^- + K_t y_t \tag{7.17c}$$

$$P_t = (I - K_t H) P_t^- \tag{7.17d}$$

Predicted (prior) estimates are marked with the superscript -, updated (posterior) estimates are marked with the superscript +. P is the state error covariance. y_t is the measurement residual, i.e. the difference between the measurement z_t and the estimated measurement $H\hat{x}_t^-$. Multiplying the residual with the Kalman gain K_t provides the correction $K_t y_t$ to the predicted estimate $\hat{x}_t^-$. [26]

The filter is initialized with an initial guess of the state estimate $\hat{x}_0^+$ and the error covariance matrix, along with noise estimation variables R and Q.

Kalman filter for linear tracking

In the case of 1D linear tracking with constant acceleration, x^t will be a vector containing object's midpoint coordinate, velocity and acceleration: $\vec{x^t} = \langle x^t, v^t, a^t \rangle$. [48] Using Newton's equations of motion, the state transition matrix is defined:

$$F = \begin{pmatrix} 1 & \delta t & \frac{1}{2}\delta t^2 \\ 0 & 1 & \delta t \\ 0 & 0 & 1 \end{pmatrix} \tag{7.18}$$

with δt being the sampling interval. [57] In the case of a point-only-measured system, where only the position and not the velocity of the object is measured, $H = \langle 1, 0, 0 \rangle$. This is easily extended to higher dimensions by extending the state transition matrix with copies of F along the diagonal and updating H to reflect more measurements. Q and R are usually not known and can be used as tuning parameters. The initial error covariance guess P_0^+ is usually large in order to ensure a quicker convergence. [26] As the initial speed and acceleration of the fish are unknown, the filter is initialised with the midpoint of the first observation of the fish and speed and acceleration set to zero. This means that the estimated velocity will be wrong for a number of frames before the filter stabilizes.

Velocity vector comparisons

The 3-dimensional velocity vectors $\vec{v_i} = \langle v_{x_i}, v_{y_i}, v_{z_i} \rangle$ and $\vec{v_j} = \left\langle v_{x_j}, v_{y_j}, v_{z_j} \right\rangle$ for two observations may be compared simply by calculating the absolute difference between the two them:

$$D_{\vec{v_i},\vec{v_j}} = \sqrt{(v_{x_i} - v_{x_j})^2 + (v_{y_i} - v_{y_j})^2 + (v_{z_i} - v_{z_j})^2} \tag{7.19}$$

As the speed of the fish generally changes more rapidly than its general direction, it may be prudent to consider the angle $\theta_{i,j}$ between the vectors as well. A large angle indicates a large difference in the directions of the fish.

$$\theta_{i,j} = cos^{-1}\left(\frac{\vec{v_i} \cdot \vec{v_j}}{\|\vec{v_i}\|\|\vec{v_j}\|}\right) \tag{7.20}$$

7.3.3 Length estimate

The length calculation itself will be discussed the next chapter. After a tracklet has been generated, we have a number of length estimates equal to the number of observations in the tracklet. By considering a high length estimate percentile, following the logic that the length is quite likely to be underestimated due to contortions of the fish, but seldom overestimated, a single length estimate is returned for each tracklet. These length estimates could be useful for tracklet linking, but relies on the fish being in a good position for length calculation at least once per tracklet, which is not always the case.

Chapter 8

Length estimation

The fish length is estimated by calculating the length of a line segment between two real world coordinate points (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) on the fish; one placed at the tip of the head and one at the end of the tail. The real world coordinates of these points are calculated as described in Section 3.1. For the sake of simplicity, the fish is always assumed to be straight when the length is calculated.

When the fish's velocity vector is not perpendicular, or almost perpendicular, to the optical axis, the length estimate may be less accurate due to parts of the fish no longer being visible. Tracking the fish across multiple frames increases the likelihood of observing the fish in a good position, i.e. relatively straight and with a direction perpendicular to the optical axis.

8.0.1 End point coordinates selection

The pixel coordinates (x_1, y_1) and (x_2, y_2) should be carefully selected to actually represent extreme points along the fish. After a binary mask is produced, a bounding ellipse is fitted to each probable fish shape. The ellipse was chosen rather than a rotated bounding box as it fits the general shape of the fish better. The bounding ellipse is defined by its midpoint coordinate (x, y), the length of the major and minor axes and the orientation of the ellipse with regards to the x-axis. The end point pixel coordinates can be determined by finding the intersections between the edge of the fish and a line along the major axis of the ellipse.

An ellipse can be represented by the implicit form $F(\vec{a}, \vec{x}) = 0$, where

$$\begin{aligned} F(\vec{a}, \vec{x}_i) &= A_{xx}x_i^2 + A_{xy}x_iy_i + A_{yy}y_i^2 + A_x x_i + A_y y_i + A_0 \\ &= \left[x_i^2, x_iy_i, y_i^2, x_i, y_i, 1\right] \cdot \left[A_{xx}, A_{xy}, A_{yy}, A_x, A_y, A_0\right] \\ &= \chi_i \cdot \vec{a}. \end{aligned} \tag{8.1}$$

$\vec{x}_i = (x_i, y_i)$ denotes a 2D data point. $\vec{a}$ contains constants which determine the shape, size and orientation of the ellipse. Given a set of data points $P = \{\vec{x}_i\}_{i=1}^n$ belonging to a detected fish, the goal is to find the best fitting ellipse to this data. The ellipse parameters are found by minimizing the algebraic distance $\epsilon^2(\vec{a})$; the sum of the distance from every point (x_i, y_i)

in P to the curve defined by the parameter vector $\vec{a}$. [16] If all points fit perfectly on the the ellipse, the error will be zero.

$$\epsilon^2(\vec{a}) = \sum_{i=1}^{n} F(\vec{a}, \vec{x}_i)^2 = \|D\vec{x}\|^2 \tag{8.2}$$

where

$$\|D\vec{a}\|^2 = \vec{a}^T D^T D\vec{a} \tag{8.3}$$

and $\|\vec{a}\|^2 = 1$. D is an $n \times 6$ matrix with rows χ_i. Using a Lagrange multiplier λ, a constrained objective function E is defined: [16]

$$E = \vec{a}^T D^T D\vec{a} - \lambda(\vec{a}^T\vec{a} - 1) \tag{8.4}$$

The constrained objective function is minimized to find the solution $\vec{a}_{min}$:

$$\nabla_{\vec{a}} E = 2D^T D\vec{a} - 2\lambda\vec{a} = 0 \tag{8.5}$$

$\vec{a}_{min}$, the best fitting ellipse to the set of points P, is the eigenvector corresponding to the smallest eigenvalue of $D^T D$ and can be found by solving the eigensystem using Hessenberg reduction and QR-factorization. [16]

Determining the fish's distance to the camera

The distances z_1 and z_2 at the end points are difficult to determine. The distance measurements at the end points frequently resemble background distance values due to possible segmentation errors and noise in the depth map. Therefore, z_1 and z_2 should be estimated using several points along the major axis of the fish, or alternatively all points on the fish.

Chapter 9

Validation metrics

9.1 Segmentation

The classification results were evaluated using four criteria: accuracy, total misclassification error, the Jaccard index and boundary F1 score. The first three are based on the amount of correctly and incorrectly segmented pixels, while the boundary F1 score evaluates how closely the contour of the segmented objects resembles the ground truth contour.

A high value returned for a single validation metric is not necessarily an indication of a successful validation. Additionally, lower results for other validation metrics, more specifically the boundary F1 score and the Jaccard index is expected and somewhat acceptable as a perfect border segmentation for every fish in every frame is difficult and likely not needed for this thesis.

Pixel-based validation metrics

Accuracy refers to the amount of correctly classified object pixels compared to the true object area:

$$acc = \frac{TP}{TP + FN} \tag{9.1}$$

TP refers to the number of true positives (pixels correctly classified as fish) and FN is the number of false negatives (pixels incorrectly classified as background). This metric should give an idea of how much of the target object is correctly identified.

The total misclassification error (ME) is the ratio of correctly classified pixels.

$$ME = \frac{TP + TN}{N} \tag{9.2}$$

TN refers to the number of true negatives (correctly classified background pixels), and N is the total number of pixels in the image. Note that ME is not independent of the size of the objects of interest, meaning that for small objects no attempt at segmentation could still result in a seemingly good ME value (since TN would be a high).

The Jaccard index, also known as intersection over union, was chosen due to its popularity as a similarity measure for two sets. It is defined as the ratio of the union of correctly classified object pixels over the intersection of ground truth object pixels and the model's predicted object pixels.

$$jacc = \frac{TP}{FP + TP + FN} \tag{9.3}$$

The Jaccard index is closely related to another popular segmentation validation metric: the Dice Coefficient. The Dice coefficient D can be calculated from the Jaccard index J using the following formula: [43]

$$D = \frac{2J}{1 + J} \tag{9.4}$$

Due to the high positive correlation between these two metrics, only the Jaccard index was used in this thesis. Note that although the Dice coefficient is also known as the F1 score, this metric is not the same as the boundary F1 score described in the following section; the boundary F1 score is a contour-based, not pixel-based, validation metric.

Contour-based validation metrics

The methods above do indicate the success of the segmentation. However they do not give much information about how closely the segmented boundary resembles the true boundary. Since accurate segmentation of the fish's extremities will help the length estimation, the boundary F1 score (BF1), introduced in [11], was also included as a segmentation validation metric. It gives an indication of the fraction of the pixels in the predicted boundary that are within a certain distance of the ground truth boundary. It is defined as the harmonic mean between modified precision P_c and recall R_c:

$$P_c = \frac{1}{|B_p|} \sum_{z \in B_p} [[d(z, B_g^c) < \theta]] \tag{9.5}$$

$$R_c = \frac{1}{|B_g|} \sum_{z \in B_g} [[d(z, B_p^c) < \theta]] \tag{9.6}$$

B_p and B_g represent the predicted object countour and the ground truth object countour respectively, and θ is the distance error tolerance set to 75% of the object diagonal. $[[z]]$ is the Iverson bracket; $[[z]] = 1$ if z is true, and 0 otherwise.

$$BF1 = 2 * \frac{P_c * R_c}{P_c + R_c} \tag{9.7}$$

The Hausdorff distance is another contour-based metric, often used in segmentation challenges. The Hausdorff distance H is the maximum of the (Euclidean) distance from a point in any of the contours to the nearest point in the other contour. [55] It is defined:

$$H = \max\left\{ \max_{a\in A} \min_{b\in B} d(a,b), \max_{b\in B} \min_{a\in A} d(b,a) \right\} \tag{9.8}$$

where A and B are the sets of all the points in each contour, and $d(a,b)$ represents the Euclidean distance from point a to point b. The Hausdorff distance is highly sensitive to outliers; a single FP far from the true contour will cause a large Hausdorff distance. This problem can be alleviated by only considering a fixed percentile f of the distances, ensuring that f% of the points are within the returned Hausdorff distance. [23] That being said, in instances with large amounts of noise, the partial Hausdorff distance will not give an indication of the portion of outlier points and the portion of points relatively close to the ground truth border. As some of the segmentation methods are expected to return large amounts of segmentation noise, but also segment the fish relatively well, the boundary F1 score was chosen over the Hausdorff distance to evaluate the segmented contour.

9.2 Tracking

As the segmentation success is evaluated separately, the correctness of the tracker is evaluated using only the fish's midpoints. A successful tracker should be able to track the fish as long as it is in the frame and produce only one track per fish. The tracklet generator and the tracklet linker will be evaluated differently. In the case of the tracklet generator, it is acceptable that it ends tracklets if the fish is lost for even one frame and it should start new tracklets in splitting and merging situations. The tracklet linker should be able to handle occlusions and short-term loss of detections due to segmentation errors.

Consider a ground truth track (GT) and a system track (ST). [73] Some error in midpoint position is expected and acceptable; we consider a target $o \in GT$ and an object hypothesis $h \in ST$ at time step t to be a valid correspondence if they are within a certain distance from each other: [5]

$$C(o,h) = \begin{cases} 1, & \text{if } dist(o,h) < T_d \\ 0, & \text{otherwise} \end{cases} \tag{9.9}$$

Since a single ground truth track may be assigned to multiple system tracks due to ID swaps or missed detections, a tracker-to-target mapping is calculated for every time step using the Hungarian method described in Section 7.1.2 with Euclidean distance as the cost function.

If, for a hypothesis $h \in ST$ at time step t, $dist(h,o_j) \geq T_d, \forall o_j \in GT_t$, the hypothesis is considered a false positive. Similarly, if a target $o \in GT_t$ does not have a valid correspondence to any hypothesis, a false negative is counted. [41] An identity swap (ID swap) is counted when a target is matched with a hypothesis with a different ID than the previous correspondence. [41] For the tracklet linker, it is important to keep the number of ID swaps low; this is less important for the tracklet generator.

Using the tracker-to-target mapping, the following tracking validation metrics can be calculated.

Multiple object tracking precision

Multiple object tracking prediction (MOTP) evaluates how precisely the tracker estimates the object's position, i.e. the target's midpoint. It is defined as the sum of the distance d_i between all corresponding object-hypothesis pairs i, averaged by the number of correspondences. [41]

$$MOTP = \frac{\sum_t \sum_i d_i}{\sum_t C_t} \tag{9.10}$$

C_t represents the number of correspondences in time step t.

Multiple object tracking accuracy

Multiple object tracking accuracy (MOTA) is one of the most widely used multiple object tracking validation metrics. [41] It is derived from three error ratios: the ratio of false positives, the ratio of false negatives and the ratio of ID swaps in the sequences. Summed up, these ratios gives us the total error rate. [5]

$$MOTA = 1 - \frac{\sum_t FP + FN + IDSW}{\sum_t GT} \tag{9.11}$$

GT describes the number of ground truth objects, targets, in each frame. FP and FN represent the total number of false positives and false negatives respectively, and $IDSW$ represents the number of ID swaps.

Part III

Experimental Results and Discussion

Chapter 10

Experimental Results

This chapter contains experimental results, presented together with their related discussions. It is divided into four main sections. The first three sections concerns the testing of segmentation, tracking and length estimation algorithms on the 3 Cod dataset. The most promising algorithms are then applied to the Fish Schools dataset in Section 10.4.

10.1 Evaluation of segmentation methods

All segmentation methods described in Chapter 6 were tested using the hand-segmented validation data set consisting of 93 frames extracted from the 3 Cod dataset. The segmentation validation metrics described in Section 9.1 are used to evaluate the results.

Note that the depth map was adjusted to contain values in the $[0, 255]$ range by multiplying every entry with $-1 \times \frac{255}{700}$ (700 cm being the theoretical largest distance from the camera) in order to utilize efficient image processing functions from the Python library OpenCV. Experiments using functions that accept any range of values showed that this conversion to 8-bit gray-scale images had no major effect on the final results. The conversion replaces None-values in the depth map with zero. This was judged acceptable, as the amount of None-entries is very low in this dataset (about 0.025% of the total number of entries in the dataset).

Box plots are used in this section to illustrate the segmentation validation metrics results. The box plots extend from the first quartile to the third quartile, with the median marked with a line. The whiskers extend to the most extreme data point within 1.5 $\times$ the interquartile range distance from the first and the third quartile. Outlier data points beyond the whiskers are marked with circles.

Grid searches were performed to determine the optimal combination of input parameters for several methods in this section. The result of a grid search is presented as a heat map with the mean validation metrics results for each parameter combination tested.

10.1.1 Global thresholding

Global segmentation was attempted on the intensity image and the depth map using the methods of Otsu and Kittler-Illingworth. While these methods are very popular, they did not produce good results in this case. The intensity of the fish varies a lot, and the fish are not necessarily brighter than the background. This, in addition to some noise in the image, makes intensity-based segmentation very difficult. The depth map has a horizontal gradient as well as noise. This caused the global segmentation methods to fail on the depth map as well.

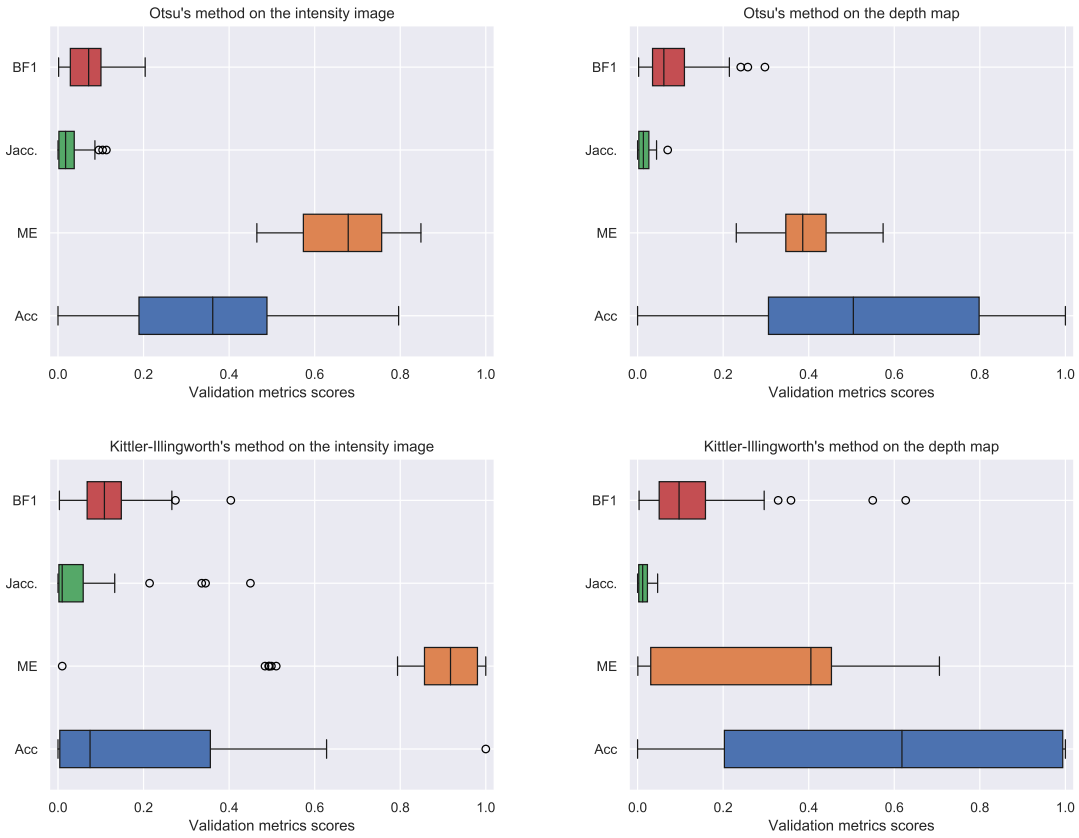


Figure 10.1: Global segmentation on the depth map and the intensity image with the methods of Otsu and Kittler-Illingworth. The low mean error scores in most of the box plots indicate that these methods produce a large amount of false positives. When Kittler-Illingworth's method was applied to the intensity image, the mean error score improved, but the low accuracy indicates that this method completely fails to segment the fish.

As the plots in Figure 10.1 illustrates, the median mean error scores were less than 80% for almost all methods, indicating a large amount of false positives. The one exception, Kittler-Illingworth's method on the intensity image, has a very low accuracy; the fish were hardly segmented. The wide boxes and long whiskers in the box plots show that the success

of the methods vary widely from image to image. After investigating the images that produced the lowest scores for these methods, it becomes apparent that the methods struggle particularly with images devoid of (or nearly devoid of) fish.

The accuracy for both methods increases on the depth map compared to the intensity image. The intensity of the fish varies widely as it changes direction, and in some cases the fish appears darker than the background making segmentation on the intensity image difficult. When the methods were applied to the depth map, the fish were more consistently marked as foreground objects, but due to the gradient background, other parts of the image were marked as foreground as well. While the validation metrics results of the global segmentation methods were poor, they indicate that we should focus on the depth map rather than the intensity image.

10.1.2 Local thresholding

Due to the gradient background in the depth map, local thresholding methods may be better suited to segment the fish.

Adaptive mean and Gaussian thresholding

Initial tests showed that basing the threshold on the local mean value or the weighted sum of the local neighbourhood produced very similar results. Adaptive Gaussian thresholding was selected as the noise particles appeared to be slightly less concentrated compared to adaptive mean thresholding.

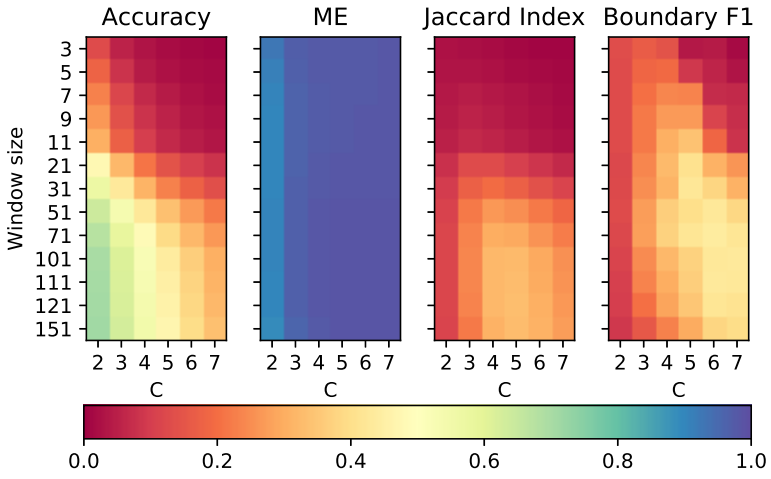
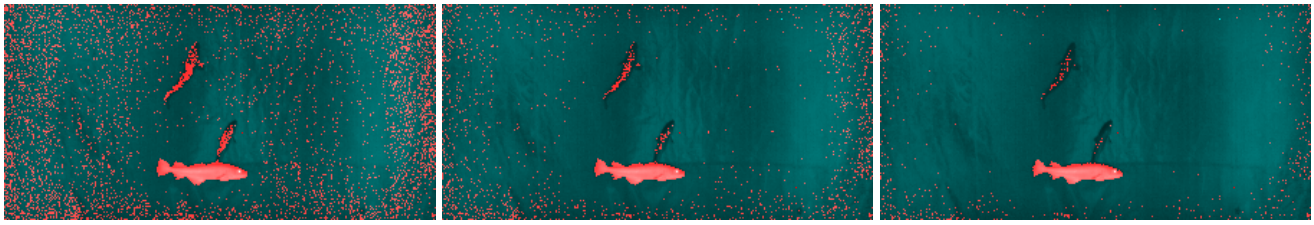


Figure 10.2: Visualized average validation metrics results for various combinations of window size and C, the two input parameters to the adaptive Gaussian thresholding method.

A grid search was performed in order to determine the best input

parameters (window size w and a constant C, described in Section 6.1.2). Figure 10.2 shows improved average accuracy and mean error scores compared to global segmentation as the window size increases. Increasing the constant C improves the average Jaccard index and boundary F1 scores by decreasing the number of false positives.

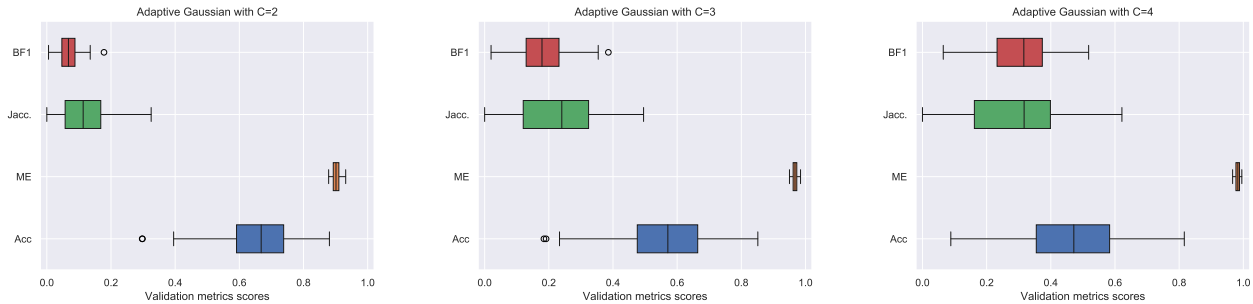
Unfortunately, no single combination of window size and C maximised the score for every metric. It seems clear that larger window size is necessary to obtain a decent accuracy score. A larger C leads to loss in accuracy, but gains for all other metrics (due to reducing the number of false positives). Investigating the effect of changing the constant C (Figure 10.3 with corresponding validation metrics scores in Figure 10.4) reveals that increasing C decreases the segmentation noise, but also removes parts of the fish segmentation.



(a) C = 2 *(b) C = 3* *(c) C = 4*

Figure 10.3: Changing the constant C reduces the accuracy due to some parts of the fish no longer being segmented, but improves all other validation metrics results as the segmentation noise decreases. The window size was set to 71.

From Figure 10.3, it seems that a substantial amount of the error stems from a large number of small false positives, largely located around the image edges. Unlike previous attempts with global thresholding, these "noise particles" are small enough that they may be removed using morphological operations. If a large enough area of the fish is correctly segmented, i.e. C is kept low enough, the actual foreground objects should not be too affected by this operation.



(a) C = 2 *(b) C = 3* *(c) C = 4*

Figure 10.4: The decrease in false positives, as seen in Figure 10.3, is reflected in the increase of the median Jaccard index, boundary F1 score and mean error score.

Niblack's thresholding method

Niblack's local thresholding method requires two parameters to be determined by the user: the window size and a constant k. A grid search was performed in order to determine the best input parameters.

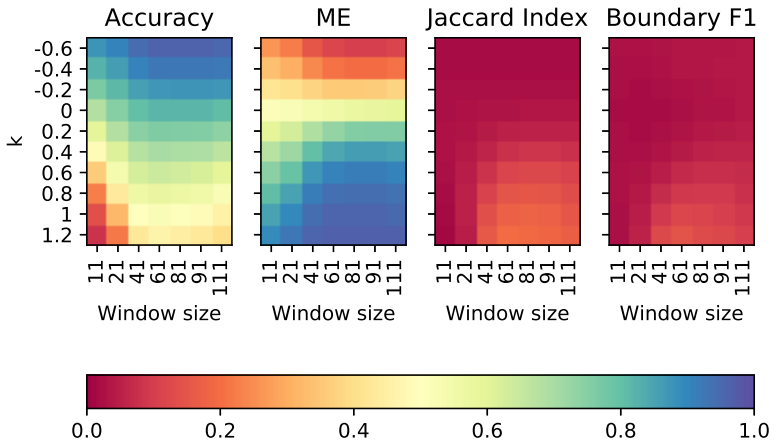
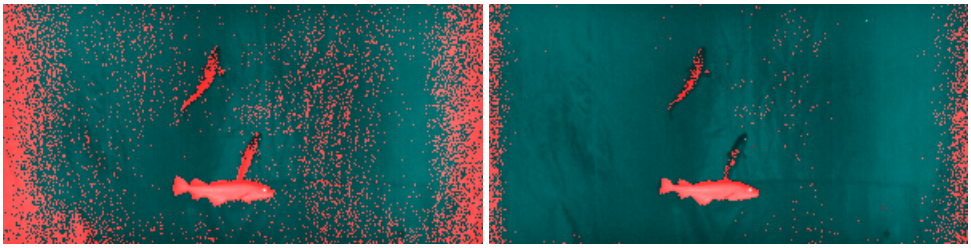


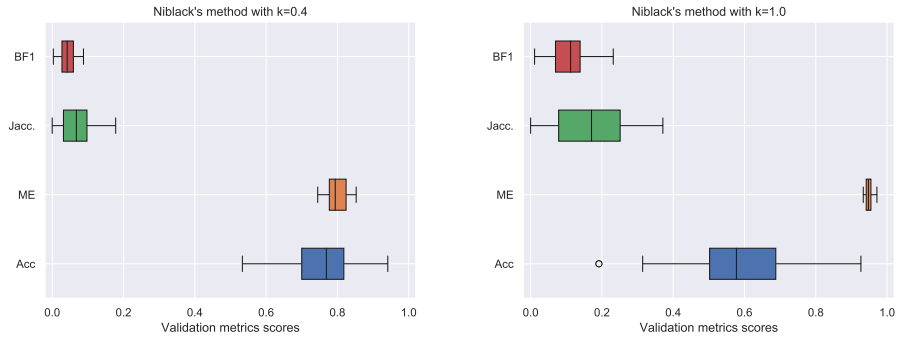
Figure 10.5: Visualized average validation metrics results for various combinations of window size and k, the two input parameters to Niblack's method.

Figure 10.5 shows that Niblack's suggested $k = -0.2$ returns low average mean error results for all window sizes tested. This indicates a large presence of false positives. Increasing k reduces the accuracy, but increases the results of the other metrics. A window size of 41 or more combined with a k in the range $[0.4, 1.0]$ may reduce the amount of false positives sufficiently so that they may be removed at a later stage, while keeping enough of the fish segmentation intact. This is admittedly a large range, and should be investigated further in conjunction with noise reduction methods.



(a) k = 0.4 *(b) k = 1.0*

Figure 10.6: Example images showing how changing k in Niblack's method affects the segmentation. Increasing k decreases noise, but the segmented object area is also decreased. The window size = 71 in both examples.



(a) k = 0.4 *(b) k = 1.0*

Figure 10.7: Validation metrics for Niblack's method with varying k and constant window size = 71. As k increases, the decreased noise level results in improved validation metric scores, with the exception of accuracy as the segmented object area decreases.

Figure 10.6, along with the corresponding validation metrics box plots in Figure 10.7, illustrates how the amount of noise varies with k. Similarly to the adaptive Gaussian method, most of the noise is placed around the image edges. However, the noise is more concentrated compared to the adaptive Gaussian method; this may make it more difficult to remove in a subsequent processing step.

Sauvola's thresholding method

As mentioned in Section 6.1.2, Sauvola's method does not generally perform well on images where the difference between the foreground intensity and the background intensity is relatively small. However, with histogram equalization prior to the segmentation, this method did return some worthwhile results. The gray-level standard deviation R was set to 128 in accordance with the author's advice, leaving us to explore window size and k.

Figure 10.8 indicates that the number of false positives decreases with an increasing k, and once again this decrease is at the cost of a lowered accuracy. Figure 10.9 illustrates how the segmentation is affected by an increasing k. Once again most of the false positives are situated near the borders.

Bataineh's thresholding method

Finally, Bataineh's local thresholding method was investigated. This method has the advantage that only one parameter needs to be set: the window size. Supposedly it also solves the problems with the methods of Niblack and Sauvola: false positives when the window passes over empty areas and difficulties dealing with small differences between the foreground and the background. Unfortunately, it quickly became clear that the output from this method contains too much noise to be useful.

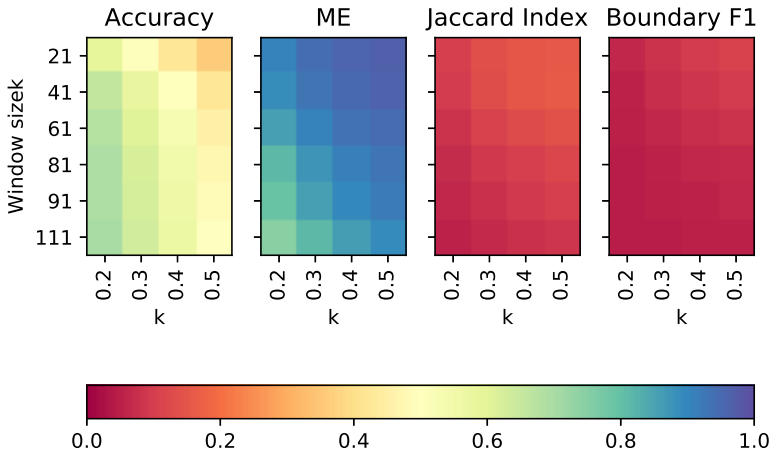
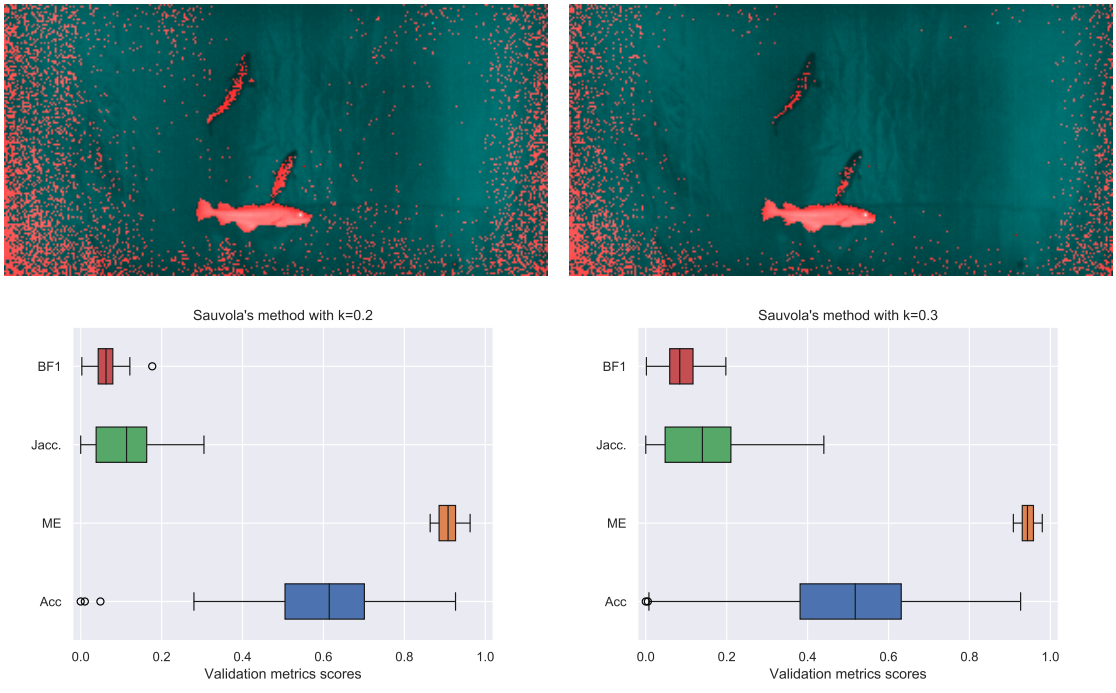


Figure 10.8: Visualized average validation metrics results for various combinations of window size and k, the two input parameters to Sauvola's method.

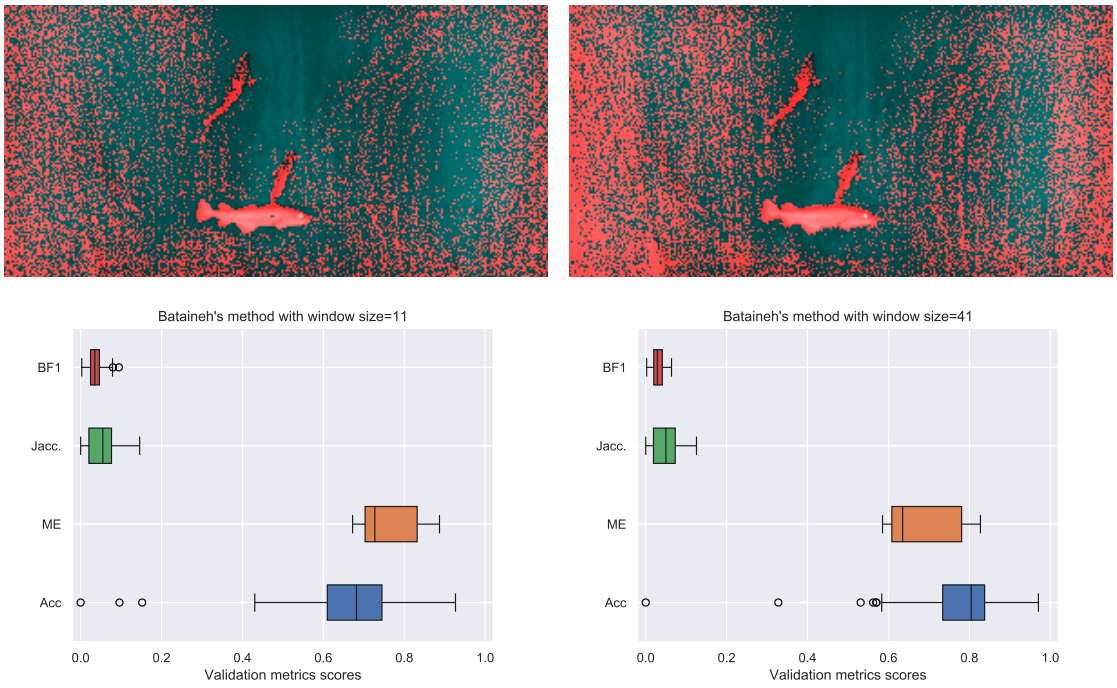


(a) k = 0.2 *(b) k = 0.3*

Figure 10.9: Validation metrics scores for Sauvola's method with varying k and constant window size = 21. Similar to Niblack's method, an increased k results in a decreased amount of segmentation noise, but also a decreased accuracy.

Figure 10.10 shows the large amount of noise in the segmented images. The noise did not significantly decrease when the window size was changed.

The high concentration of the noise makes it difficult, if not impossible, to remove. Therefore Bataineh's method was not used moving forward.



(a) window size = 11

(b) window size = 41

Figure 10.10: Bataineh's method failed to produce adequate results, with large amounts of noise regardless of the window size. The high accuracy scores in the box plots indicate that the fish are mostly segmented, but the results of the other validation metrics are relatively low.

10.1.3 Segmentation by background subtraction

If the background of the depth map is relatively smooth and homogeneous, it may be modelled as described in Section 6.2. Once a function has been fitted to the depth map, a background model can be subtracted from every frame. Every pixel for which the difference between the current frame and the estimated background is larger than a threshold T_D is assumed to belong to a fish. The depth map background modelling process includes some parameters that may require tuning: degree of the fitted function, the number of points used to fit the function, and the difference threshold T_D.

Using 50 random points in the function fitting and a difference threshold of 3 pixels, the effect of changing the degree d of the polynomial function modelling the background (see equation 6.18) was investigated. As the depth map has been converted to an 8-bit image, a threshold $T_D = 1$ pixel is approximately equivalent to $2.75 \approx \frac{700}{255}$ cm. Figure 10.11 shows that while there are only minor changes in the validation metrics scores as the degree increases, it seems that degree $d = 3$ offers an improvement in mean error compared to degree $d = 2$. The difference between degree $d = 3$ and

degree $d = 4$ seems negligible. In order to avoid over-fitting, degree $d = 3$ was used moving forward.

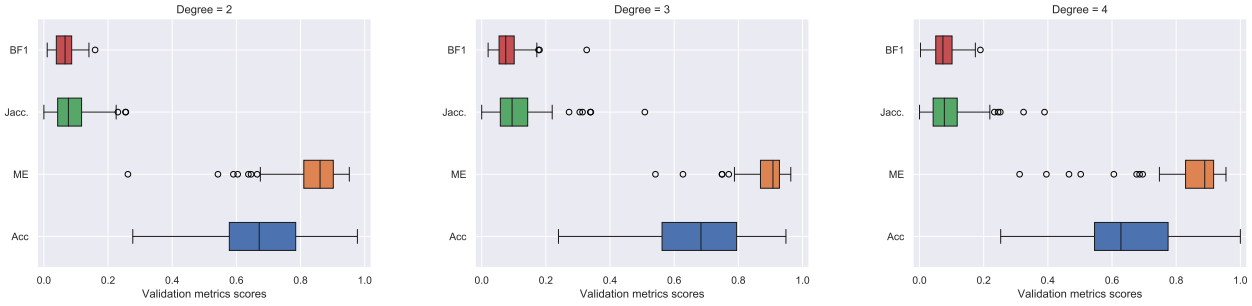


Figure 10.11: Validation metrics results of various degrees of the polynomial function used to model the background. Increasing the degree only slightly improves the result.

Next, a suitable number of points for the fitting of the polynomial function must be selected. The function is faster to compute with a smaller number of input points, but it may not be accurate enough. A high number of points will create a more accurate model, but the model may also be over-fitted. Figure 10.12 shows that using more than 50 random points hardly improves the segmentation.

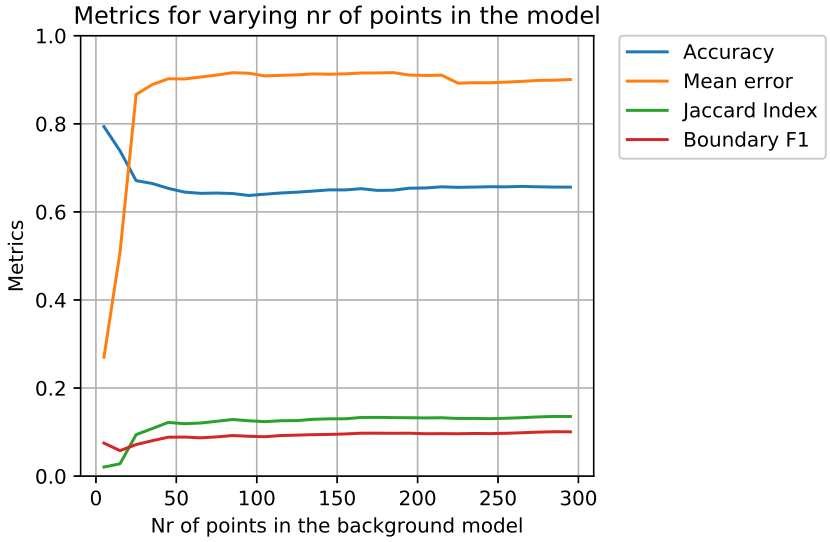
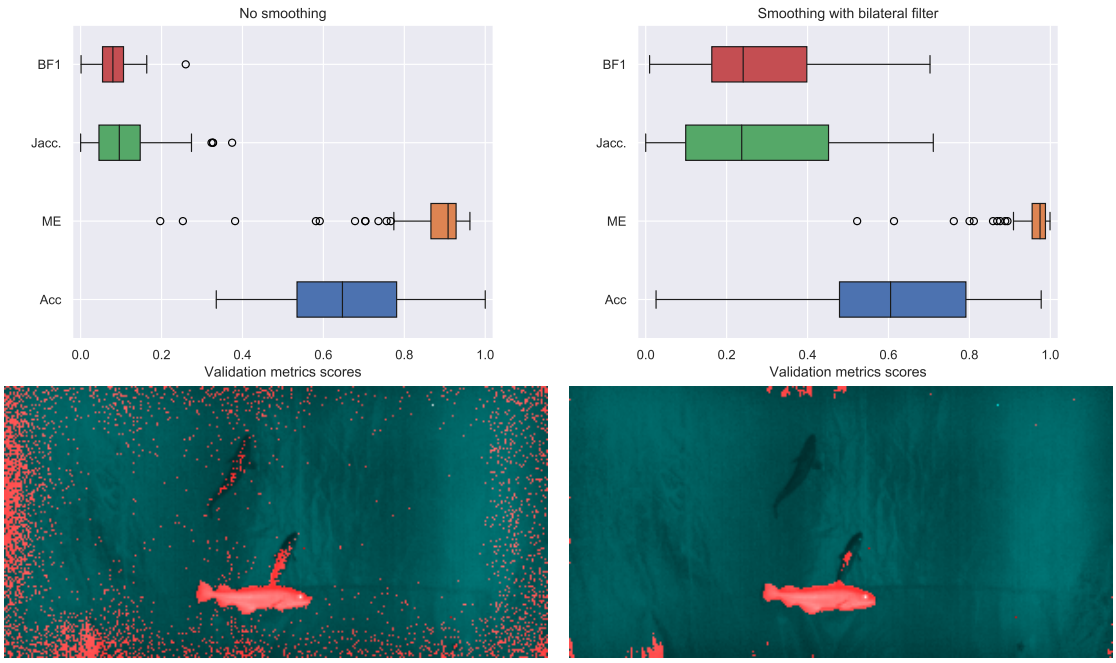


Figure 10.12: Number of points used in depth map background estimation. There is little improvement in the results of the performance metrics results for more than 50 points.

If the depth map background is not relatively smooth, the fitted function will not be able to model the background accurately enough. A bilateral filter (as described in Section 6.4.1) was applied to the depth map prior to the fitting in the hope that it would improve model by smoothing

the background while preserving the edges of the fish. Figure 10.13 shows that applying the smoothing filter does remove noise. All validation metrics results are improved except from accuracy (as it removes parts of the object segmentation as well).



(a) No smoothing *(b) Bilateral Filter*

Figure 10.13: Deph map model with degree $d = 3$ based on 50 randomly selected points and a difference threshold $T_D = 3$. The bilateral filter size was 7×7, with $\sigma_r = \sigma_s = 30$. Applying the bilateral filter to the depth map prior to estimating the background model removes a large amount of noise. However, one of two partially segmented fish in (a) is not detected in (b), and the segmentation of the other is much poorer.

A more suitable difference threshold T_D may help keep the object segmented and still remove most of the noise. Figure 10.14 displays the averaged validation metrics results for various thresholds and degrees of the fitted function. It is clear that when the mean error increases, most likely due to false positives being removed, the accuracy decreases. Keep in mind that a high mean error is achievable by no segmentation whatsoever, as the foreground objects are small compared to the background.

An important thing to note in Figure 10.13 is the large interquartile range and the high number of outliers in the box plots, which indicates that this segmentation method does not return consistent results. The background model is based on a number of randomly selected points, and this random point selection process is likely the cause of the high variability in the results. The hope was that since the foreground is small compared to the background, the majority of these points would belong to the background and when foreground points were inevitably included,

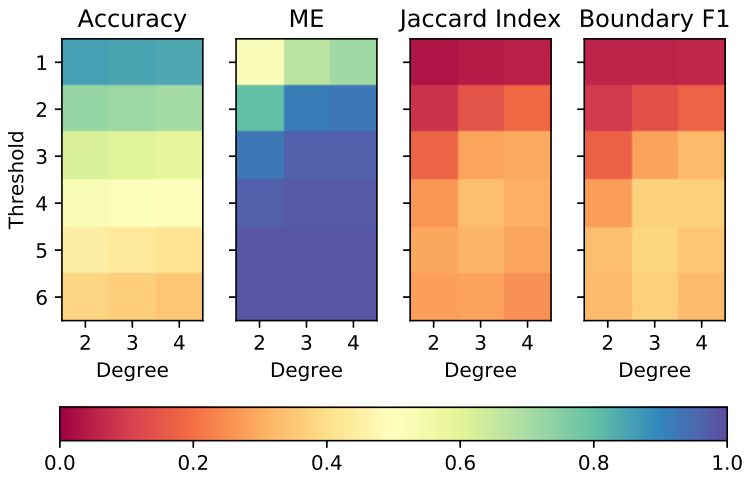


Figure 10.14: Averaged validation metrics results for various combinations of difference thresholds and polynomial degrees. The depth map was smoothed using a bilateral filter prior to the segmentation.

they would not disturb the model too much. However, the high variability indicates that the random selection affects the model greatly. Figure 10.15 shows how the random point selection causes the background subtraction segmentation to output three very different segmentation results for the same input image.

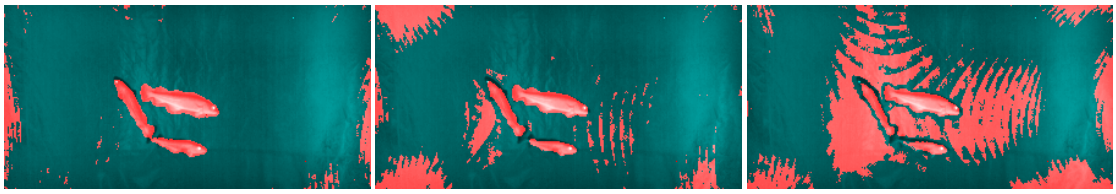


Figure 10.15: Even with the same input parameters, the randomness in the point selection process produces very different results; from near-perfect to unusable.

Assuming that even in poorly segmented frames, the amount of true positives stays high and that the randomness results in varying amounts and positions of false positives, an iterative solution may be viable. By calculating and subtracting the depth map multiple times for every frame and labelling pixels as fish only if they belong to the set of the most frequently segmented pixels, a reliable and accurate segmentation may be achieved.

Figure 10.16 indicates that such an algorithm depends on how often false positives are produced, as it will still fail if one area is continuously mislabelled as fish. The number of iterations should be kept as low as possible for the sake of computational speed while still be high enough for the randomization to return some well-segmented frames. The threshold

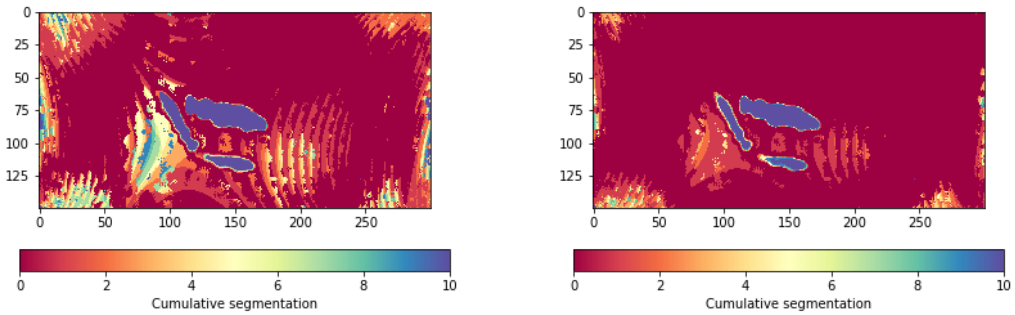
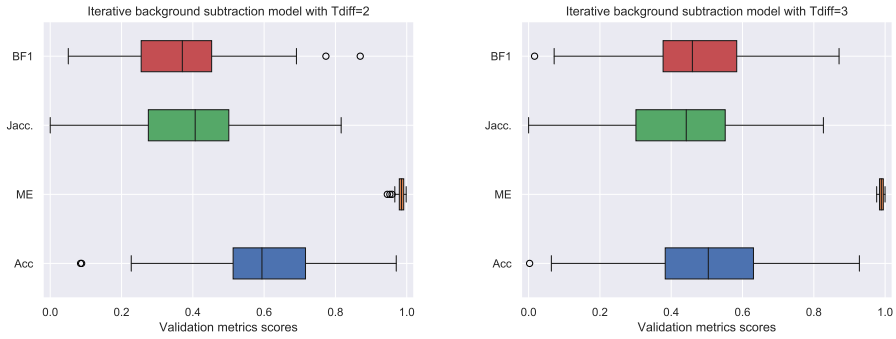


Figure 10.16: Heatmaps showing the number of times each pixel is marked as foreground after ten iterations using a difference threshold $T_d = 2$ (left) and $T_d = 3$ (right). The majority of the fish is segmented in every iteration.

T_D must be carefully selected so that that the number of false positives is minimized and the fish is still consistently segmented.

Figure 10.17 illustrates the validation metrics results for an iterative background subtraction method with ten iterations. All pixels labelled as fish at least nine times are assumed to belong to the foreground. The validation metrics results have improved from Figure 10.13. That being said, this segmentation method is much less computationally efficient than the global and local threshold methods described previously.



(a) $T_d = 2$ *(b)* $T_d = 3$

Figure 10.17: Validation metrics results for the iterative background subtraction method with ten iterations, accepting all pixels labelled as foreground at least nine times. Compared to Figure 10.13, all metrics results except accuracy are improved for $T_D = 3$, indicating a reduced number of false positives.

10.1.4 Noise removal

Of the methods tested in this section, adaptive Gaussian and iterative background subtraction returned the most promising results. Granted, adaptive Gaussian had a large amount of false positives if the C-variable was kept low enough to segment most of the fish, but the false

positives were mainly small, separated particles that should be easily removable using morphological operations. The iterative background model subtraction returned less segmentation noise and better validation metrics scores than the adaptive Gaussian thresholding, but the noise particles were generally larger and more difficult to remove. Additionally, this segmentation method was significantly slower than all other methods tested in this section. It seems likely that adaptive Gaussian thresholding can return equivalent or better validation metrics results after noise removal.

Morphological opening is an operation that removes small components from a binary image. As long as the segmented components of the fish are larger than the noise particles, the opening should not disturb the segmentation of the fish significantly apart from smoothing the edges. A high accuracy was prioritized as it gives a good indication of how much of the fish is correctly segmented. From Figure 10.4, a high window size seems to increase the accuracy up to a point. Therefore, window size $w = 111$ in the following calculations. The constant C was kept low, again to keep the accuracy high.

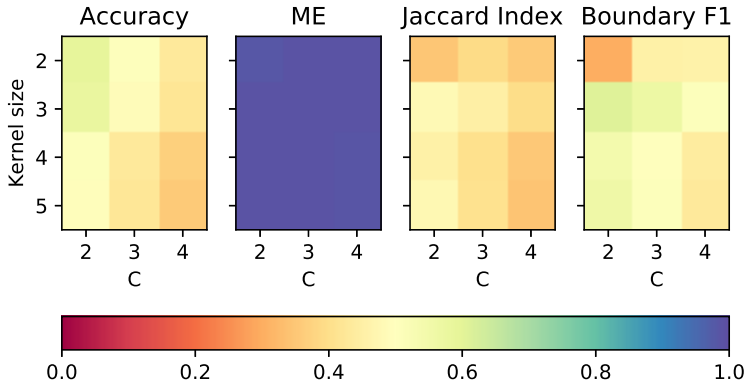


Figure 10.18: Grid search to determine the kernel size of a square structuring element for noise removal. It seems that a 3×3 structuring element combined with $C = 2$ returns high accuracy and improved Jaccard index and boundary F1 scores.

The size and shape of the structuring element determines the size and shape of the noise particles it can remove. It is important to choose a structuring element that removes a large amount of the noise while keeping the true fish segmentation intact. A grid search for the size of the structuring element and a good corresponding C was performed. Figure 10.18 shows that a 3×3 square structuring element combined with $C = 2$ returns high mean accuracy and improved the mean Jaccard index and

boundary F1 scores.

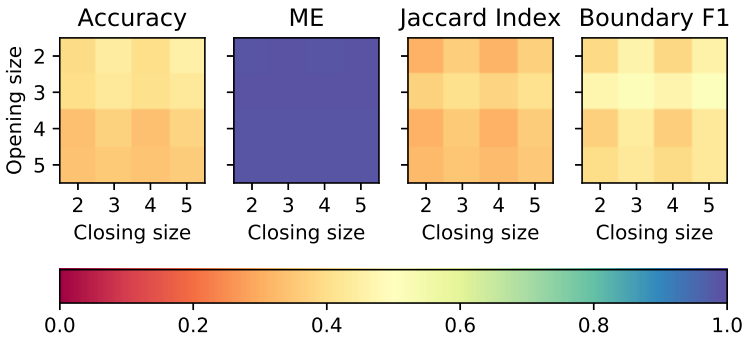


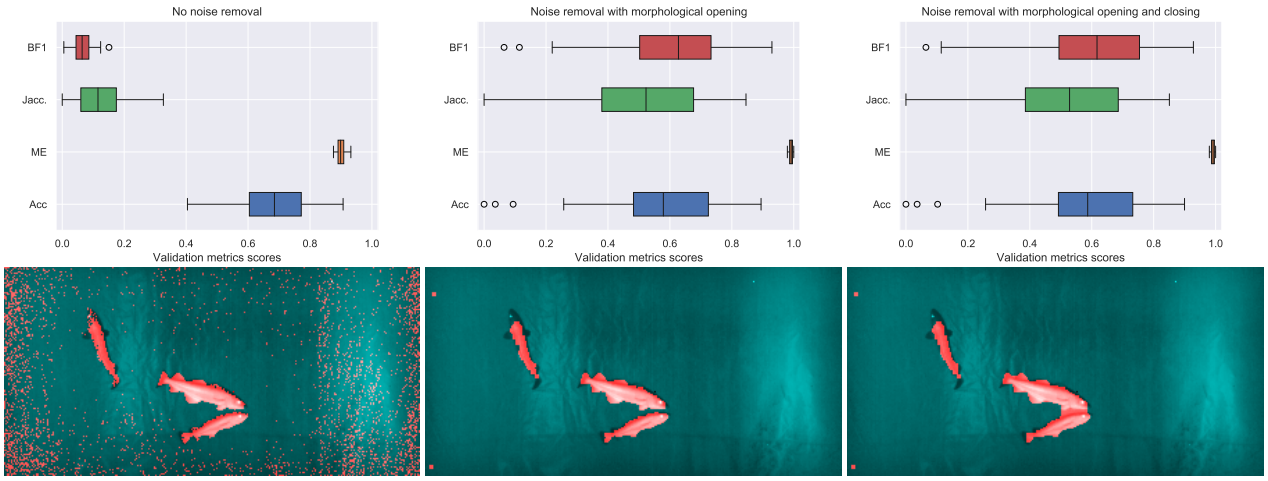
Figure 10.19: Grid search for the ideal size of the structuring elements used for morphologcial opening opening and closing. It seems that a 3×3 *opening element pairs well with a* 3×3 *or* 5×5 *closing element.*

The possibility that morphological closing would further improve the results by closing potential holes created by the opening was also investigated. Figure 10.19 indicates that the 3×3 opening element for the opening pairs well with either the same closing element or a 5×5 closing element. Investigating these two options reveals that closing with a 5×5 structuring element yields a slightly higher accuracy at the expense of a slightly lowered boundary F1 score.

Figure 10.20 shows the effect of morphological opening and closing on an image from the segmentation validation set. Before the morphological opening, there is a large amount of small noise components around the edges of the image. After morphological opening, almost all of these noise components have been removed and the Jaccard index along with the boundary F1 score have been vastly improved. We accept a small drop in accuracy; this is due to small, disconnected segmentation components in the fish being removed. The morphological closing improves the accuracy marginally at the expense of border accuracy (we see for instance that the two lower fish have merged into one observation).

10.1.5 Edge detection

While the adaptive Gaussian thresholding and iterative background modelling returned relatively good validation metric results, the edges of the fish are often poorly segmented (as an example, consider the leftmost fish in Figure 10.20, which lost parts of its tail in the noise removal). Detecting edges and adding them to the segmented image may improve the segmentation.

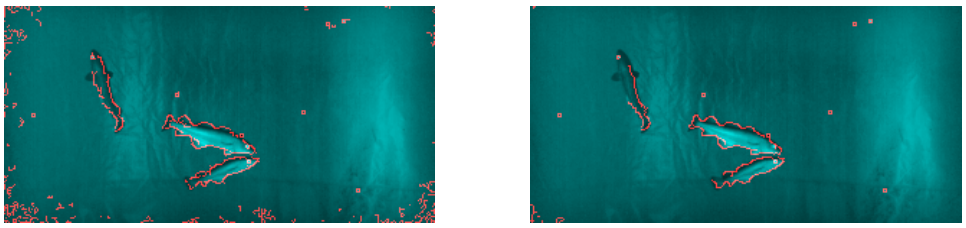


(a) Before noise removal *(b) After morphological opening* *(c) After morphological closing*

Figure 10.20: Noise reduction with morphological closing and opening, both with a square 3×3 structuring element.

When applying the edge detection to the depth map, only edges belonging to the fish should be returned, as the depth map background in the 3 Cod dataset contains no other abrupt transitions. A bilateral smoothing filter was applied to the depth map prior to the edge detection to smooth small errors and noise. The effect of the bilateral filter is demonstrated in Figure 10.21. The detected edges are added to the binary mask returned by the adaptive Gaussian threshold.

For Canny's edge detection, the lower and upper thresholds T_L and T_U need to be determined by the user. Low thresholds mark more of the fish edges, but also return a higher number of false positives. Several thresholds were tested; Figure 10.21 shows that $T_L = 10$ and $T_U = 20$, along with the bilateral filter, marked most edges on the fish with minimal noise.



(a) No bilateral filter *(b) Bilateral filter*

Figure 10.21: Applying a bilateral filter prior to the edge detection significantly reduces the amount of noise returned.

Canny's edge detection combined with adaptive Gaussian thresholding followed by morphological opening was tested for various structuring element sizes. To avoid the edges being removed by the morphological opening, the edge detection output was dilated before being added to the adaptive Gaussian thresholding mask.

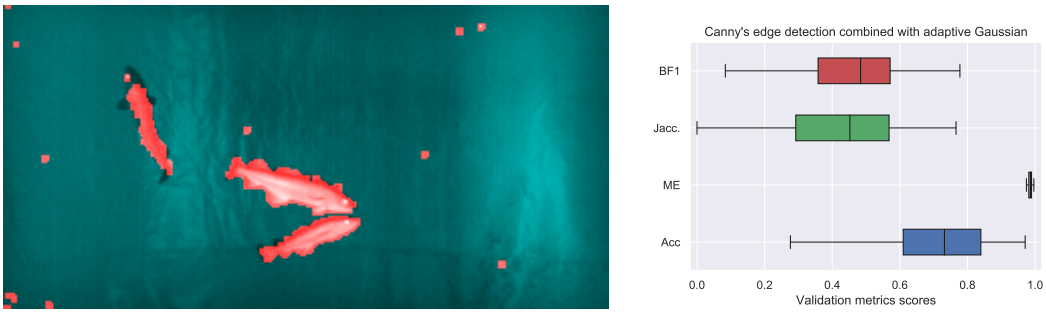


Figure 10.22: Canny's edge detection, dilated with a 2×2 elliptical structuring element, combined with adaptive Gaussian thresholding with window size 111 and $C = 2$. Noise was removed with a 3×3 square structuring element. Comparing the validation metrics results with the results in Figure 10.20, the edge detection does improve the accuracy at the cost of adding false positives, evident in the decreased boundary F1 and Jaccard index scores.

While adding the edge detection did improve the accuracy, the improvement was accompanied by an increase in noise, as illustrated by Figure 10.22. Most of the false positives added by the edge detection are relatively small and can likely be removed at a later step by checking the sizes of all connected components. However, the benefits of adding the edge detection was judged to be minor compared to the detriments of possibly adding false detections. Additionally, Canny's edge detection will likely produce a larger number of false positives on other datasets, such as placing an edge at the transition between None-entries and background entries in the Fish Schools dataset. Therefore, Canny's edge detection was not used moving forward.

10.1.6 Segmentation summary

Due to high variability in the intensity of the fish (it may be lighter or darker than the background), segmentation on the intensity image was discarded. Instead, the fish were segmented using the depth map, where the fish are generally closer to the camera than the background. Global segmentation yielded poor validation metrics results, but the local thresholding methods all performed better. Of the four local thresholding methods tested, adaptive Gaussian thresholding with window size=111 and $C = 2$ will be used moving forward, as it returned a near 70% median accuracy and the segmentation noise consisted mainly of small particles. While morphological opening and closing with a square 3×3 structuring element reduced the median accuracy to 59%, it removed most of the noise particles and vastly improved the other metrics (the median mean error rose from 90% to 99%, the median Jaccard index rose from 0.12 to 0.53 and the median boundary F1 score rose from 0.06 to 0.62).

Iterative segmentation by background subtraction on the depth map returned promising results. However, this method is slow compared to the thresholding methods tested and it did not outperform adaptive Gaussian

with morphological noise removal. Edge detection with Canny's edge detector did pick out the edges of the fish. Combing the results with the output of the adaptive Gaussian segmentation did improve accuracy, but not the other validation metric results. As the edge detection has the potential to introduce false observations in the tracking due to some larger noise particles, it was not used in the final algorithm.

In conclusion, adaptive Gaussian thresholding seems likely to detect most fish while not introducing many false detections that may confuse the tracker. A perfect segmentation of each detection was deemed an unreasonably difficult task; the adaptive Gaussian thresholding seems capable of producing near-perfect segmentation in many instances where the fish is well-positioned (i.e. relatively straight and close the camera). This should be sufficient to return a good length estimate, as the calculations are based on multiple observations of the same fish.

10.2 Cost features and tracking results

10.2.1 Cost features

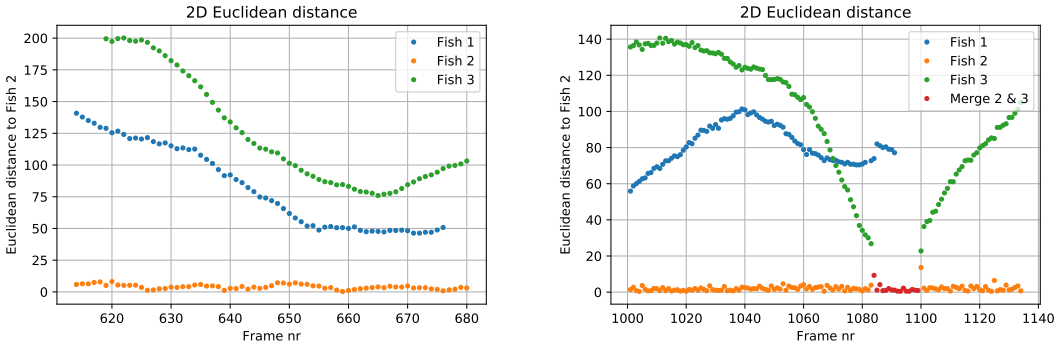
In order to link different observations of the same fish, a selection of features that describe the similarity between the two observations must be found. The requirements for these features are different for the tracklet generator and the tracklet linker; for the tracklet generation we are only concerned with consistency between subsequent frames, while in the tracklet linking process consistency over several frames is essential.

Using validation data sets where each fish $x_j^t, j = 1, 2, 3$, observed at time t, is marked with a unique ID, the cost features were tested by comparing each observation to a fish with a predetermined ID in the previous frame. For the feature to be useful in the tracking algorithm, it should return lower scores when x_i^{t-1} is compared with x_i^t rather than $x_j^t, j \neq i$. In all the following examples, each fish $x_j^t, j = 1, 2, 3$ in frame t of the validation data sets is compared to x_2^{t-1} (Fish 2) in the previous frame $t - 1$.

Euclidean distance

An obvious cost feature is the center coordinate of an object. Since the video has a high frame rate, the fish will not move much between frames and unless an occlusion occurs, an observation should usually be linked to the closest observation in the previous frame.

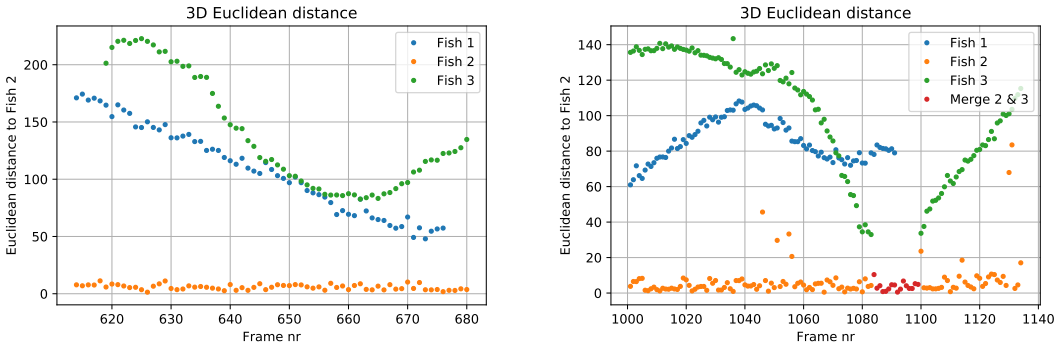
Figure 10.23 shows that Euclidean distance using only the (x, y)-coordinate of a fish's midpoint separates the fish clearly. It also marks merge and split situations well, as the distance to any object at these points are larger than usual. In Figure 10.24, the midpoint distance z was added. This did not make this cost feature more reliable. For 3-dimensional Euclidean distance to be used as a cost feature, the fish's position may need



(a) No merging *(b) Merging*

Figure 10.23: The Euclidean distance from the midpoint (excluding the midpoint distance z) of every fish to the midpoint of Fish 2 in the previous frame. Unsurprisingly, the distance is always smallest when comparing Fish 2 with Fish 2. (b) displays a merge situation. The merge and the split is clear in the plot; the distance from the merged observation in red to Fish 2 around frame 1082 and the distance from Fish 2 to the merged observation at frame 1000 is unusually large.

to be smoothed with a Kalman filter prior to the cost calculation. This will hopefully remove some outliers due to, for instance, segmentation errors.



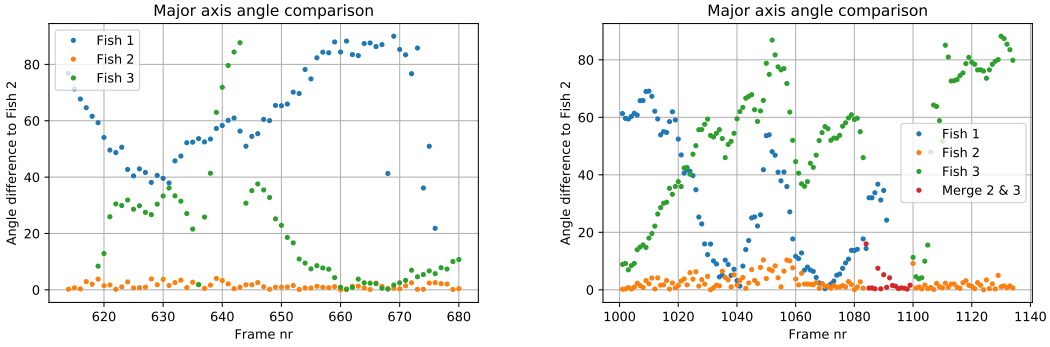
(a) No merging *(b) Merging*

Figure 10.24: The Euclidean distance from the midpoint (including the midpoint distance z) of every fish to the midpoint of Fish 2 in the previous frame. Adding the midpoint distance does not make this cost feature more reliable. In some cases, the midpoint is placed outside the fish as the fish bends or due to segmentation errors, causing jumps in distance.

Major axis angle

The angle of the major axis may help to differentiate fish swimming close to each other. For frame-by-frame tracklet generation, no attempt at orientation estimation or head-tail detection was made, so the maximum angle difference would be 90 degrees for two fish whose major axes are

perpendicular to each other. Figure 10.25 shows that the major axis angle does work as a cost feature for most frames; it generally returns low results when comparing x_2^{t-1} and x_2^t as the fish's orientation usually does not change much between frames. However, it does occasionally also return low results when comparing x_2^{t-1} and $x_i^t, i \neq 2$ as multiple fish are often oriented similarly with regards to the x-axis.



(a) No merging

(b) Merging

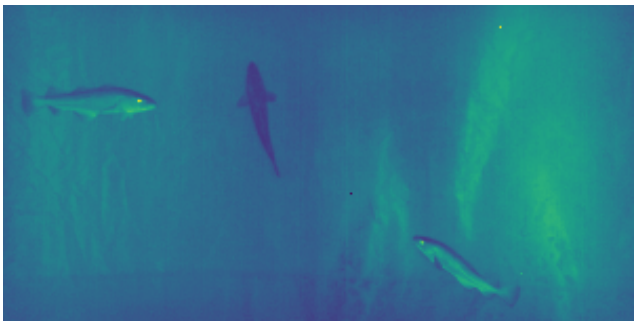
Figure 10.25: The difference in the major axis angle to the x-axis of every fish compared to Fish 2 in the previous frame. For large parts of the two sequences, this angle can be used to link together observations of Fish 2, but the costs of these links are not consistently lower than the costs of other links (for instance in (a), time steps 660 to 670).

Intensity histogram comparison

The intensity of the fish varies greatly. This caused problems when attempting global segmentation, as the fish are not necessarily brighter than the background. However, it may be useful when comparing the intensity histograms of the fish frame-by-frame. Using the segmentation mask, the intensity values at every point on the fish was retrieved. Figure 10.26 shows that the intensity histograms are clearly different for each of the three fish in the frame. If they do not change too much from frame to frame, the histograms may be the basis of a useful cost feature.

Using Minkowski distance with L_1 and L_2 as a similarity measure, the histogram difference was calculated over time for various numbers of bins. In this case, a lower number of bins seems to separate the fish better than a high number. The choice of L_1 or L_2 distance appears to affect the scale of the plots in Figure 10.27 more than the separation. All cost features will be scaled using a Gaussian function, so in reality there is little difference between the two. Figure 10.27 shows that using 25 bins and L_1 in the the Minkowski distance calculation correctly indicates which observation should be linked to the previous observation of Fish 2.

Earth Mover's Distance (EMD) was also investigated as a cost feature. EMD has an advantage over Minkowski distance in that it does not depend on the number of bins and allows partial matches. Figure 10.28 indicates



(a) Frame nr 626

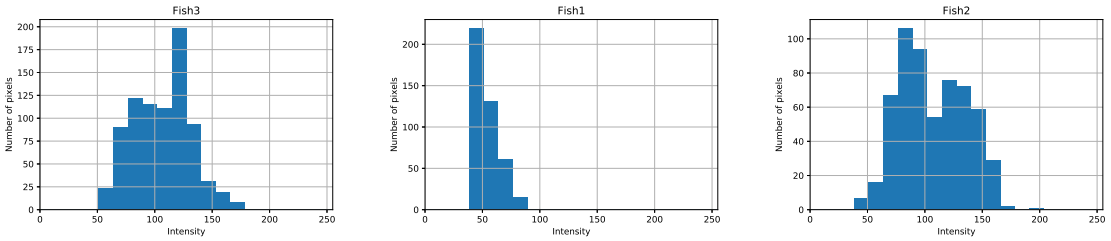
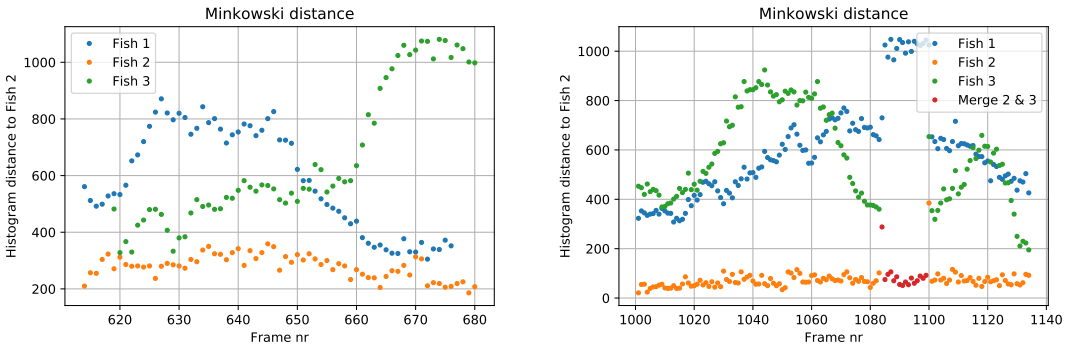


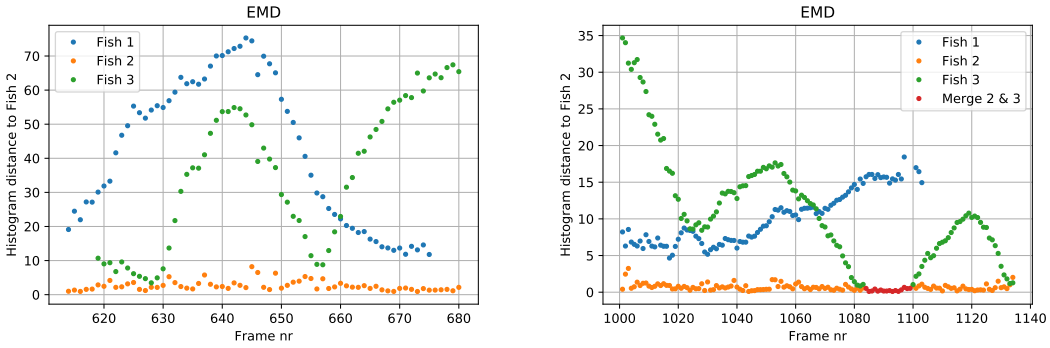
Figure 10.26: The intensity histograms of three fish in the same frame. Fish 1 (middle) is clearly darker and covers a smaller area than the two other fish. Fish 3 (left) and Fish 2 (right) seem to have similar brightness levels, but their histograms peak at different intensity values, and have different shapes.



(a) No merge *(b) Merge*

Figure 10.27: Comparing the histogram of every fish in each frame to the histogram of Fish 2 in the previous frame. With 25 bins and L_1 distance, there is a clear separation. The only exception is around frame 1082 and frame 1100, where the intensity histogram changed suddenly due to Fish 2 and Fish 3 merging.

that EMD generally returns higher values when comparing two different fish. Note that the EMD between a merged object and its components in the previous frame is small due to partial matching of the distributions (clearly seen in Figure 10.28 (b); the EMDs from Fish 2 and Fish 3 to the first merged observation of the two fish, marked in red, are small). This may be useful information at a later state when attempting to solve merge situations.



(a) No merge *(b) Merge*

Figure 10.28: Calculating the minimum cost of comparing the histogram of every fish in each frame to the histogram of Fish 2 in the previous frame. As EMD allows partial matching, the separation from the merged observation around frame 1082 to Fish 2 and Fish 3 is smaller than when using the Minkowski distance. The separation is at times poorer than the Minkowski distance results, especially around frame 628 in (a).

Velocity vector comparisons

After having used a cost feature, or a combination of cost features, to create short, connected tracklets, one may use these tracklets to extract further information about the targets, such as their velocity. A Kalman filter is applied to a fish's midpoint over time. The hope is that the Kalman filter will smooth errors in the position, especially the fish's distance, and return a velocity vector estimate for every observation.

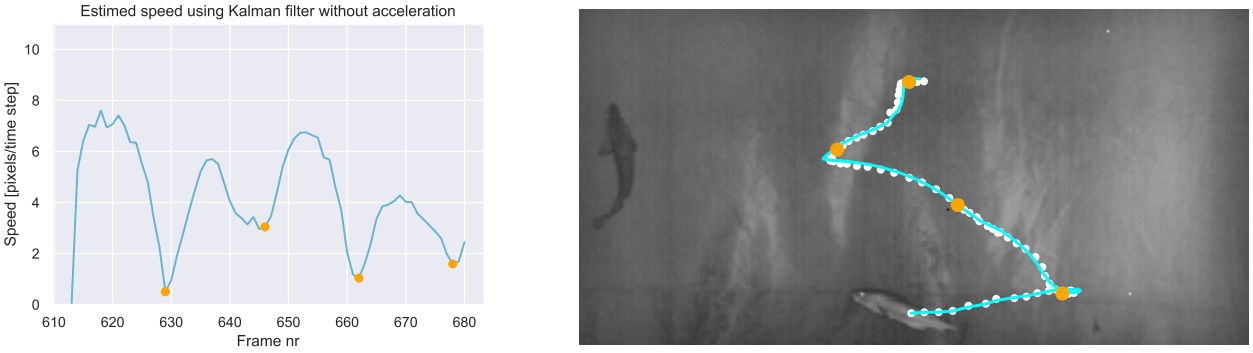


Figure 10.29: The estimated position and speed over time, with no acceleration in the Kalman filter. Orange dots indicate original measurements, the blue line is the Kalman filter output. Local speed minima are marked in the graph and in the image; they mainly correspond to changes in direction.

Initially, the Kalman filter was run without acceleration. Figure 10.29 shows that applying the filter smooths the curve and that local speed minima usually correspond to changes in direction. Adding acceleration in

the filter produced similar results (see Figure 10.30), but with different peak heights. We cannot know whether adding acceleration to the filter returns more accurate velocity estimates as the fish's true speed is unknown, but investigating the track shows that adding acceleration improved the filter lag slightly. Note that in both cases, the filter was initialized with zero velocity and zero acceleration. The filters are seemingly stabilized within 10 frames.

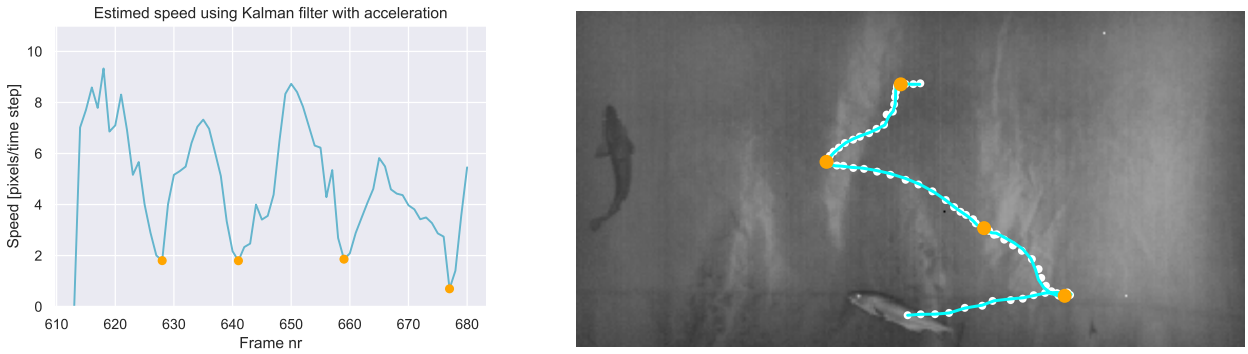


Figure 10.30: Estimated position and velocity over time with acceleration included in the Kalman filter. Note that the peaks are higher compared to Figure 10.29, where acceleration was not included

The Kalman filter returns a velocity vector estimate for every observation in a tracklet, with the exception of tracklets consisting of a single observation. If the direction of the fish does not change rapidly, calculating the difference between the velocity angles of detected fish may be useful in solving occlusions.

Figure 10.31 shows the velocity angle difference between all fish observed in time step t and the velocity angle of Fish 2 in time step $t - \Delta t$. Note that the time separation Δt is larger than one time step in these calculations, as the velocity is intended to help solve instances of occlusion or loss of detections. Neither graph shows a good separation. However, the camera moves in this particular data segment which may negatively influence the results. The same calculations were completed on another data segment, which includes a merge. Once again, the results in Figure 10.32 are not conclusive; initially there are several instances when the angle between Fish 2 and a previous observation of Fish 2, $x_2^{t-\Delta t}$, is larger than the angle between another fish and $x_2^{t-\Delta t}$, perhaps due to the Kalman filter needing time before it is stabilized. However, in this instance the angle is helpful in solving the merging of Fish 2 and Fish 3, marked with grey in the plots.

The velocity vectors themselves were also compared in a similar fashion using the absolute difference between the vectors on the same data segments. Once again, the results were inconclusive, but showed some improvement on the second merge segment with a stationary camera.

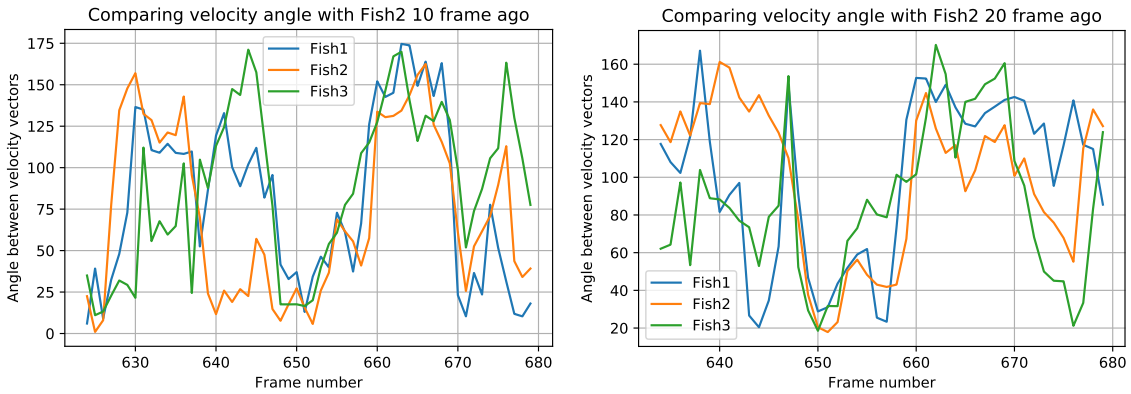


Figure 10.31: Left shows the angle between a target's velocity vector in the current frame compared to the velocity vector of Fish 2 10 frames ago. Right shows the same angle difference with a gap of 20 frames. Neither graph show a good separation when comparing the same fish and a different fish. The results are likely influenced by the camera moving in this time segment; all fish appear to have a very similar velocity. The camera is mostly stationary between frame 650 and 660; these frames in the left plot indicate that vector angle difference can be a useful cost features.

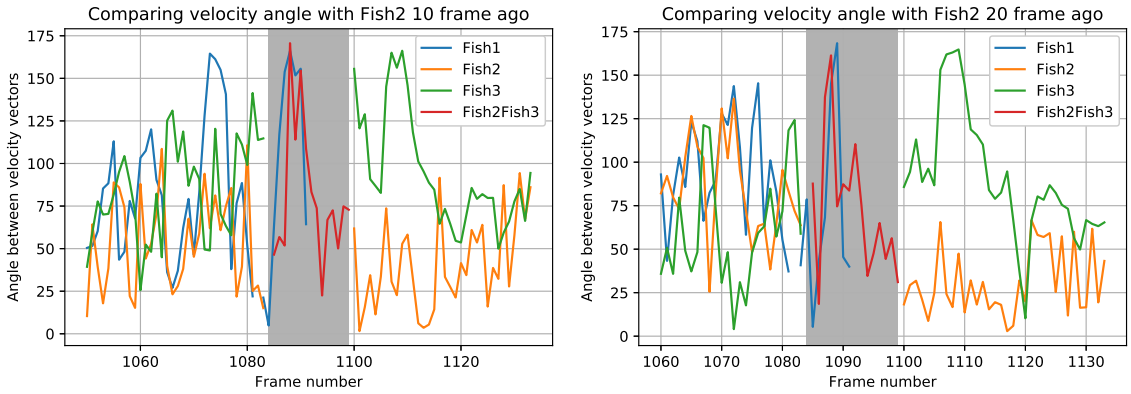


Figure 10.32: Velocity vector angle difference between each fish in the current frame and Fish 2 in an earlier frame. The merge of Fish 2 and Fish 3 is marked in dark gray. In this particular example, velocity vector angle difference would be able to solve the merge situation.

10.2.2 Tracklet Generation

Having investigated segmentation methods and cost features individually, tracklet generation may be attempted. The output of the tracklet generator is evaluated using the tracking validation dataset described in Section 4 and the tracking validation metrics described in Section 9. The fish are not tracked through occlusions; a new tracklet is initiated with the midpoint of the merged fish. After the merged fish split again, a new tracklet with a new ID is created for each fish. The goal is to create short tracklets that will be connected in a later step.

The set of tracklets generated by the algorithm (system tracklets, ST) are compared to a set of ground truth tracklets (GT) over 501 frames. An observation in ST at time t is linked to the nearest observation in GT at time t using the Hungarian method, as long as the distance between them is less than a distance threshold T_d. A 10-frame time delay was allowed in merge instances before an ID swap was counted as it became clear that system tracklets did not always merge in the same time step as the ground truth tracklets. A mere one frame delay in merging was counted as an ID swap, heavily influencing the results.

Unless the segmentation method changes, the MOTP (sum of the distances between the GT points and their matches in ST, averaged by the number of matches) will stay constant for each distance threshold T_d regardless of the cost function, as will the number of false negatives and false positives. Using the adaptive Gaussian thresholding method with window size 111, $C = 2$ and noise removal with a square 3×3 structuring element, the results in Table 10.1 were achieved.

Table 10.1: MOTP, FN and FP for adaptive Gaussian thresholding on the 501-frame test set. The MOTP indicate that the calculated midpoint is, on average, a little over 3 pixels removed from the ground truth midpoint. FN and FP decreases with an increasing T_d up to a point as more distant matches are allowed. $T_d = 20$ seems to adequately link the system tracklets to the ground truth tracklets.

Distance Threshold T_d	**MOTP**	**FN**	**FP**
10	3.17	94	43
15	3.36	75	24
20	3.42	71	20
25	3.42	71	20

Euclidean distance, major axis angle difference, Minkowski distance and EMD were investigated as potential sole cost function features. The cost features were all normalized using a Gaussian function, with a mean $\mu = 0$ and a variance σ based the maximum allowed difference between two linked observations.

Table 10.2: Tracking metrics results for the tracklet generator using a cost function based on a single cost features. Multiple σ-values were tested for each cost feature. This table presents the best MOTA results with corresponding ID swaps numbers, and the smallest σ achieving those results.

Cost feature	σ	**MOTA**	**IDSW$_\mathbf{g}$**	**IDSW$_\mathbf{h}$**
Euclidean distance	$\sigma_{dist} = 15$	0.902	4	5
Major axis angle	$\sigma_{angle} = 35$	0.897	8	8
Minkowski	$\sigma_{mink} = 400$	0.902	4	6
EMD	$\sigma_{emd} = 10$	0.898	7	2

The optimal σ for each feature was found by examining the MOTA and ID swaps for each σ. Both ST ID swaps $IDSW_s$ (a system tracklet being linked to multiple ground truth tracklets) and GT ID swaps $IDSW_g$

(a ground truth tracklet being linked to multiple system tracklets) were considered. A high $IDSW_s$ indicates that a system tracklet is not split when multiple fish merge, which makes it more challenging to solve merges at a later stage. A high $IDSW_g$ indicates poor observation linking. At this stage, poor linking is somewhat acceptable, as these poorly linked tracklets will likely be linked in the tracklet linking stage. Nevertheless, if $IDSW_g$ is too high, the tracklets will be too short to extract meaningful information from them.

The results of the σ-search can be found in Table 10.2. Euclidean distance seems to be the best sole cost feature with $MOTA = 0.90$, $IDSW_g = 4$ and $IDSW_h = 5$. The two histogram comparison methods, Minkowski distance in particular, also showed promise. Note that EMD has the lowest $IDSW_h$-rate; this indicates that this cost feature generally ends tracklets if two fish merge or split as the intensity histogram abruptly changes. Minkowski distance results for $\sigma_{mink} \leq 350$ had similar $IDSW_h$-rates, but lower MOTAs than for $\sigma_{mink} = 400$.

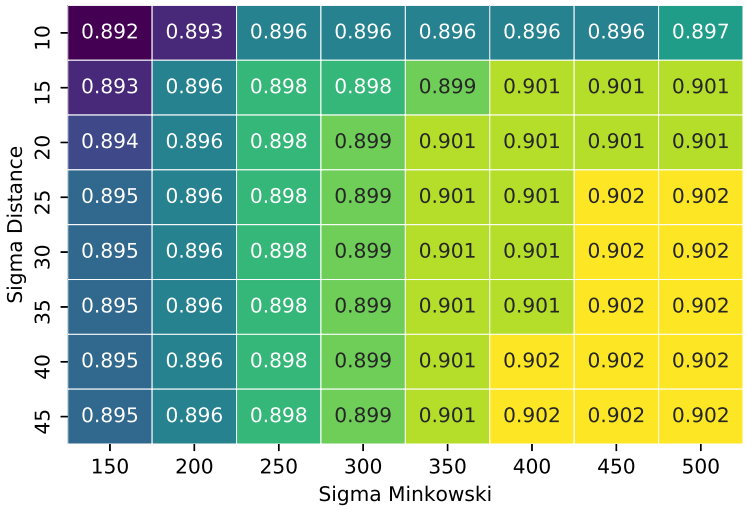


Figure 10.33: MOTA results for tracklets generated using a cost function with Euclidean and Minkowski distance and various σ_{dist} and σ_{mink}.

Hoping to decrease the number of ID swaps, Euclidean distance was combined with Minkowski distance and a grid search was performed to find the optimal variances. Figure 10.33 shows that there is no improvement in the MOTA when combining the two cost features. Figure 10.34 illustrates that there is no improvement in number of ID swaps either, compared to the best results using only Euclidean distance. The grid search was repeated for cost functions combining distance and major axis angle difference, and distance and EMD. Once again, there was no improvement compared to distance alone.

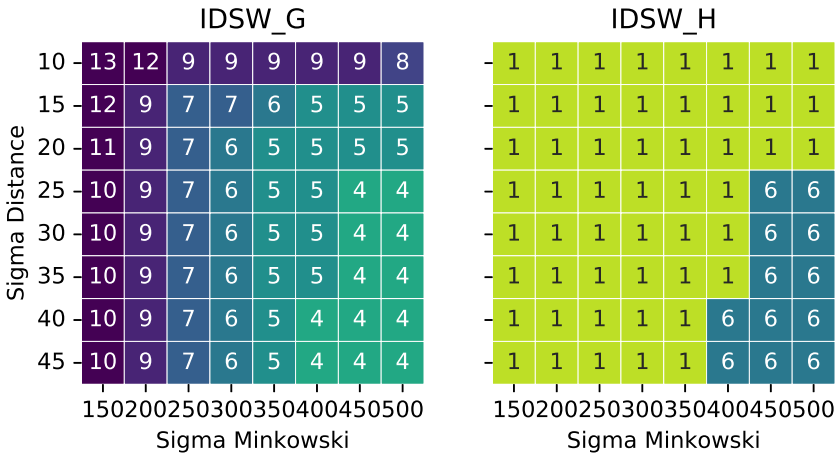


Figure 10.34: $IDSW_g$ and $IDSW_h$ rates for a tracklet generator with a cost function based on Euclidean distance and Minkowski distanc, with various σ_{dist} and σ_{mink}.

Splitting function

As a high $IDSW_h$ mostly indicates failure to split in merge situations, making it difficult to correctly identify merges and add occlusion hypotheses by copying those tracklets, an algorithm that attempts to identify merge situations and split tracklets if necessary was created. The algorithm simply searches for all instances where a tracklet terminates or initializes abruptly far from the border. For every instance, all tracklets in the subsequent (or previous) frame are split as shown in Figure 10.35.

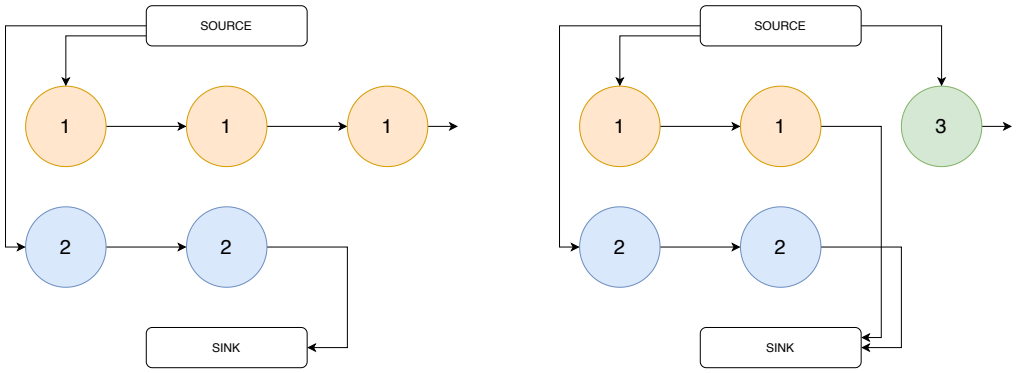


Figure 10.35: In this example, tracklet 2 terminates far from the border and an observation in tracklet 1 is sufficiently close to the last observation in tracklet 2. It is likely that parts of tracklet 1 contain merged observations of the two fish in tracklet 1 and tracklet 2. Therefore, tracklet 1 is split and a new tracklet 3 is created, initialized one time step after the final observation in tracklet 2.

Table 10.3 presents the effect of the splitting function on the validation

metrics results. MOTA decreases due to $IDSW_g$ increasing, as the splitting function splits tracklets in non-merge situations. This is expected and trivial, as these split tracklets should be easily re-connected at a later stage. However, there are no instances of a system tracklet being connected to two or more ground truth tracklets, indicating that the splitting function works as intended.

Table 10.3: MOTA and ID swaps before and after splitting tracklets. There are no more instances of a system tracklet being connected to two or more ground truth tracklets. $IDSW_g$ has unsurprisingly increased and MOTA (which depends on $IDSW_g$) has decreased due to the splitting function splitting tracklets in non-merge situations.

	MOTA	**IDSW$_g$**	**IDSW$_h$**
Before splitting	0.902	4	5
After splitting	0.894	11	0

10.2.3 Tracklet linking

After the tracklets have been created, it is possible to extract more information about the fish, most notably the fish's length and velocity, in order to connect the tracklets and create the final tracks. In the testing of the tracklet linking algorithm, all tracklets containing multiple fish were ignored due to the difficulty in assigning the correct GT tracks to the ST tracks when multiple system tracks share one node and because correctly solved merges sometimes returned unfavourable results when splits or merges occurred slightly earlier or later in the test set. Ignoring nodes with multiple fish was judged appropriate; the main issue is making sure the tracked fish keep their original ID after a merge event.

Table 10.4: The tracklet connector improves the ID swap values for the ground truth, but distance alone does not correctly resolve merge situations. The higher MOTA after splitting compared to Table 10.3 is due to tracklets containing more than two fish being ignored.

	MOTA	**IDSW$_g$**	**IDSW$_h$**	**FN**	**FP**
After splitting	0.906	9	0	50	8
$\sigma_{dist} = 15$	0.916	3	3	50	7

An initial test run using only Euclidean distance in the cost function demonstrated how the tracklet linker largely corrected the increased MOTA and $IDSW_g$ caused by the splitting function (see Table 10.4) by reconnecting split tracklets, as well as slightly improving the ratio of false positives (as the tracklet likelihood function removes unlikely detections). That being said, Euclidean distance alone cannot correctly solve merge situations; all the ID swaps in this case are due to tracklets not regaining their correct ID after a merge. The tracker produced better results when the linking of two tracklets, one terminating and one initializing in the same

frame t, was allowed, as two parts of the same fish would occasionally be detected separately in a frame.

Earth Mover's Distance, length, velocity angle and velocity vector difference were tested as sole cost functions for varying σs. The best results for the various metrics are presented in table 10.5 along with the lowest σ that achieved the result.

Table 10.5: Best MOTA results for the tracklet linker using cost functions based on a single cost feature.

Cost feature	σ	**MOTA**	$\mathbf{IDSW_g}$	$\mathbf{IDSW_h}$
Euclidean distance	$\sigma_{dist} = 5$	0.916	3	3
EMD	$\sigma_{EMD} = 5$	0.914	4	4
Length difference	$\sigma_{length} = 15$	0.916	3	3
Velocity angle	$\sigma_{angle} = 30$	0.912	4	2
Velocity vector	$\sigma_{vel.\ vector\ diff.} = 10$	0.910	5	5

Investigating the cost features individually reveals that the Euclidean distance and histogram comparison such as EMD are able to solve situations in which a few detections are missing from a track, but are unable to solve long-term loss of detections such as occlusions.

Length as a cost feature is very sensitive to the contortion and orientation of the fish. The test set includes a case in which the length of one fish involved in a merge is severely underestimated (due it's direction not being almost perpendicular to the optical axis at any point prior to the merge). Even though the lengths of the other fish involved in the merge were more accurately estimated, this resulted in all three fish swapping IDs after the merge. In this dataset, it is known that all the three fish have different lengths, but this is not necessarily true for other datasets. If several of the fish observed in a video have similar lengths, length as a cost feature is useless.

Comparing velocity angles is impossible in situations where a tracklet contains a single detection. These situations occur predominately near the edges of the image; the area of the detected fish is small as it enters or exits the frame and some detections may be discarded as noise. These situations are generally correctly resolved using features based on distance or intensity histograms. That being said, velocity angle correctly resolves all merges in the test set, including a challenging merge-and-split involving all three fish (as illustrated in Figure 10.36).

The velocity vector difference produced worse results than the velocity vector angle difference; in general failing to resolve merge situations. The velocity vector difference takes the speed of the fish, in addition to direction, into account. As the fish are changing speed rapidly and the camera is occasionally moving as well, this is less stable than comparing directions only.

As different cost features resolved different issues with the tracklet linker, a combination of cost features is likely the best option. Euclidean distance proved useful in linking tracklets with missing detections, while

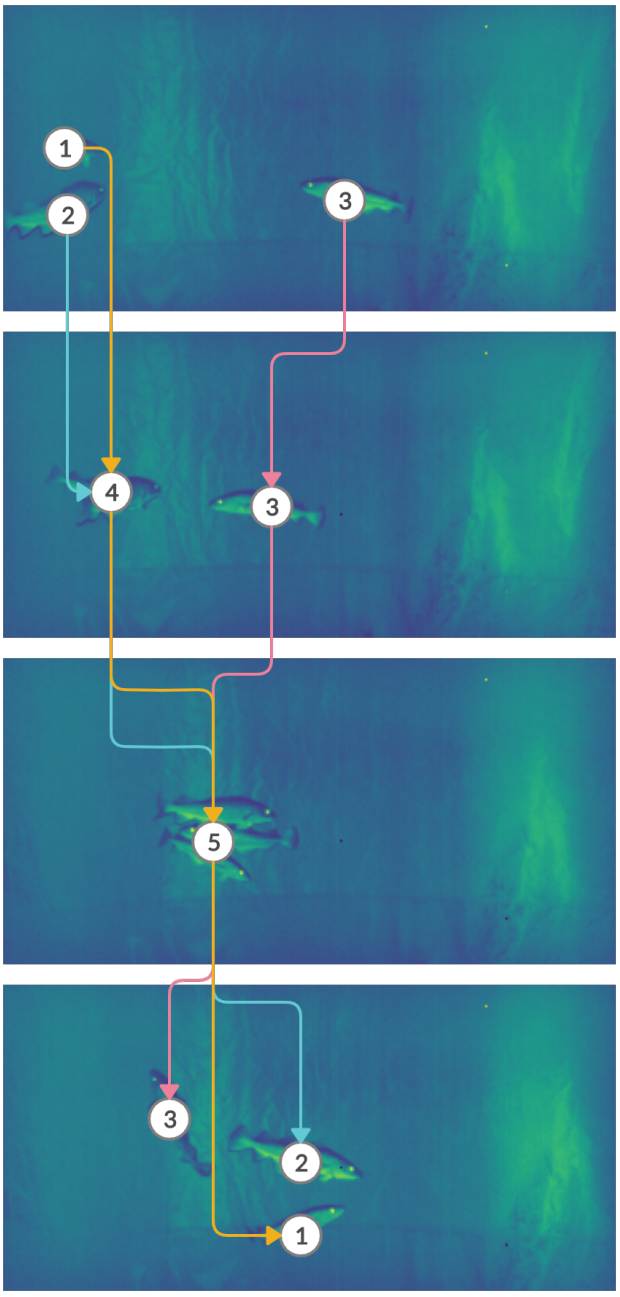


Figure 10.36: A merge and subsequent split illustrated. In this case, the IDs in the nodes indicate which track the observation belongs to, and the arcs illustrate the connections between them. Node 4 and node 5 are special cases of nodes containing more than one fish, as they have multiple incoming and outgoing arcs. Note that the direction of fish 3 is useful in recovering the right ID. Fish 2 swimming away from the camera as it comes into frame from the left makes accurate length estimation difficult. Velocity angle difference was the only single cost feature that correctly solved this merge example.

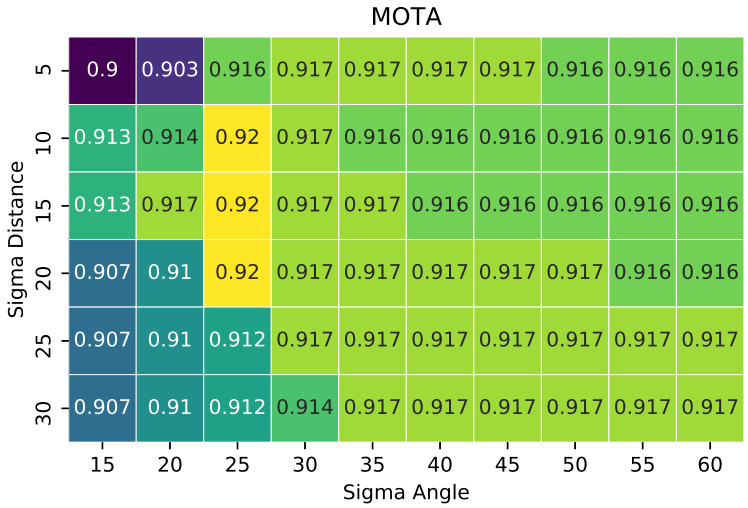


Figure 10.37: The MOTA results for the tracklet linking with a cost function based on Euclidean distance and velocity angle difference for various σ_{angle} and σ_{dist}. $\sigma_{angle} = 25$ along with $\sigma_{dist} \in \{10, 15, 20\}$ produced the highest MOTA results.

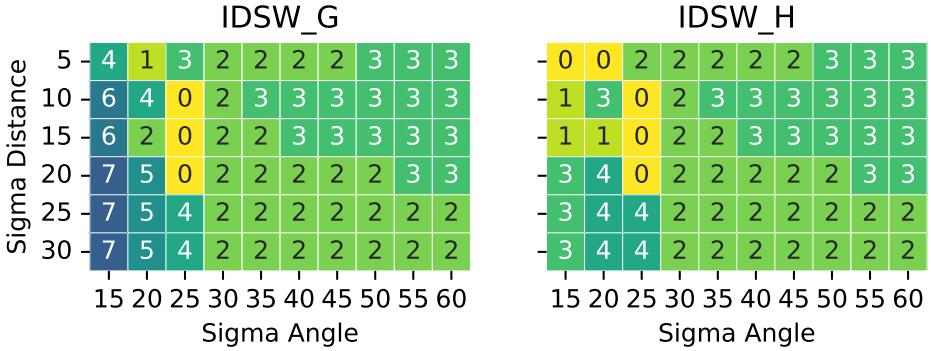


Figure 10.38: ID swaps for tracklet linking with a cost function based on Euclidean distance and velocity angle difference and various σ_{angle} and σ_{dist}. $\sigma_{angle} = 25$ along with $\sigma_{dist} \in \{10, 15, 20\}$ had no ID swaps, indicating that all fish in the test set regained their original ID after a merge.

velocity angle difference was a helpful feature in resolving merge situations. Intensity histogram comparisons, length and velocity vector difference were discarded as cost features due to instability or only linking tracklets that are also linked using angle difference or distance.

The MOTAs presented in Figure 10.37 and the ID swaps presented in Figure 10.38 shows that $\sigma_{angle} = 25$ and $\sigma_{dist} \in \{10, 15, 20\}$ correctly resolved all merge situations and linked tracklets with missing detections as well. The MOTA was increased to 0.920; this is higher than any MOTA result achieved with a single feature.

While this cost function works well on the 3 Cod dataset, it relies on the fish mostly swimming in different directions, which is not always the case. Additionally, the fish needs to be observed separately prior to and after the

merge in order to estimate the velocity. A school of fish swimming close together in the same direction would likely result in merge situations with seemingly multiple possible solutions. The algorithm connected tracklets on the 501-frame tracking validation set quickly, but as the number of fish increases, so does the complexity of the assignment problem.

Provided the number of fish is not very high, the tracklet generator can work in real time. Despite not correctly solving merges, useful data can be extracted from its output.

10.2.4 Tracking results summary

Of the four cost features tested separately for the tracklet generator, simply using the Euclidean distance returned the best combination of MOTA (0.90) and ID swaps results. The two histogram comparison methods, Minkowski distance and EMD, ended tracklets more effectively in occlusion situations. This largely ensured that each tracklet contained observations of only one fish. A similar effect was accomplished by combining the tracklets generated using Euclidean distance with a splitting function. As this approach was simple yet effective, it was used in the final algorithm. Combining Euclidean distance with a histogram difference-based comparison method did not improve the results compared to using Euclidean distance alone.

The tracklet linker required cost features that are somewhat constant across multiple frames. Using the estimated length of the fish in each tracklet seemed promising, but proved to be very sensitive. One fish being underestimated can result in multiple fishing swapping IDs, even if the other length estimates are relatively accurate. Focusing instead on the direction of the fish, comparing the velocity angle returned promising results but failed in many cases to link tracklets with only one observation. Finally, velocity angle comparison and Euclidean distance were combined to form the final cost function; Euclidean distance was able to effectively link short tracklets and velocity angle was able to correctly resolve all occlusions in the test dataset. With this cost function, a MOTA of 0.92 was achieved.

All these tests were performed on a single tracking validation dataset consisting of 501 frames. In a sense, these cost functions are specifically adapted to this dataset; there is no guarantee that they will perform equally well on other datasets. A difference in frame rate may require a different σ-value when using distance as a cost function. A dataset capturing fish mostly swimming in the same direction could render velocity angle useless as a cost function. The movement of the camera is also a concern; while the camera is mostly stationary in this dataset, this may not always be the case.

That being said, the tracking algorithm is easily adaptable to other cost functions. The tracklet linking algorithm is more likely to fail than the tracklet generator, but useful data could be extracted from the tracklets only. Average lengths could be estimated using lengths estimated from each tracklet, along with some function designed to remove tracklets likely to contain multiple fish (perhaps based on EMD, as Figure 10.28 indicates

that it is capable of identifying two fish merging).

10.3 Length estimation results

The length of the fish is estimated to be distance between two real world coordinate points on the fish. These real world coordinates are calculated using the equations in Chapter 3. The calculations require two points in the image, (x_1, y_1) and (x_2, y_2) (placed at the head and the tail of the fish, as demonstrated in Figure 10.39), and their distances to the camera z_1 and z_2.

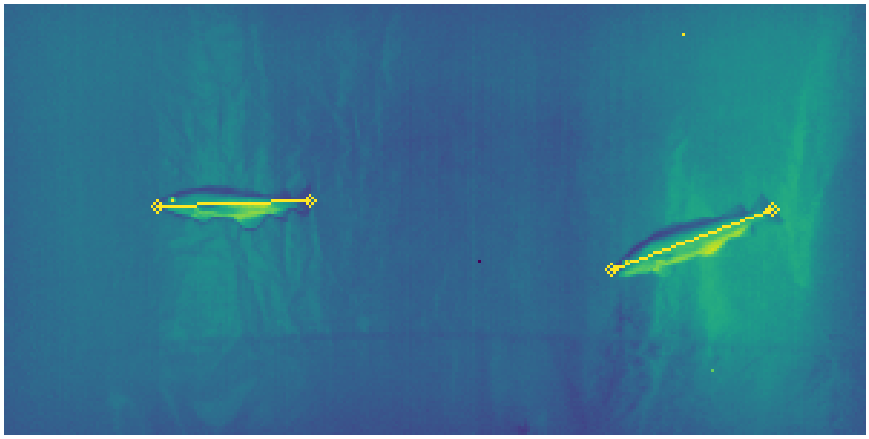


Figure 10.39: Two extreme points on the segmented fish mask are found using the major axis (marked with a yellow line in the image). The fish is, for sake of simplicity, assumed to be straight any time its true length is calculated.

10.3.1 Investigating point selection errors

Taking into account the non-linear increase in the area of view (AOV) from pixels near the image center to pixels near the image border, the length estimation algorithm calculates the AOV for each individual pixel. A linear increase would indicate that the AOV difference between two neighbouring pixels, horizontally or vertically, is always the same, while in reality this difference is slightly larger near the the image center.

Provided the fish is relatively well-positioned and well-segmented, the length estimation algorithm should return the same estimated length regardless of the fish's position in the image. In order to make sure that this was the case, the length of one fish was estimated at irregular intervals over 320 frames. Figure 10.40 shows the length plotted against the fish's distance to the image center. The estimated length of the fish varies; this is due to the fish not always being well-positioned. However, there seems to be no correlation between the distance from the center and the maximum estimated length at that distance. Since the length of the fish is more likely to be underestimated, due to segmentation errors or contortions, than overestimated, the largest calculated lengths of a track are assumed to best

reflect the true length of the fish. The relatively constant maximum length at different distance intervals from the center indicates that the length estimation algorithm is indeed stable for objects at any point in the image.

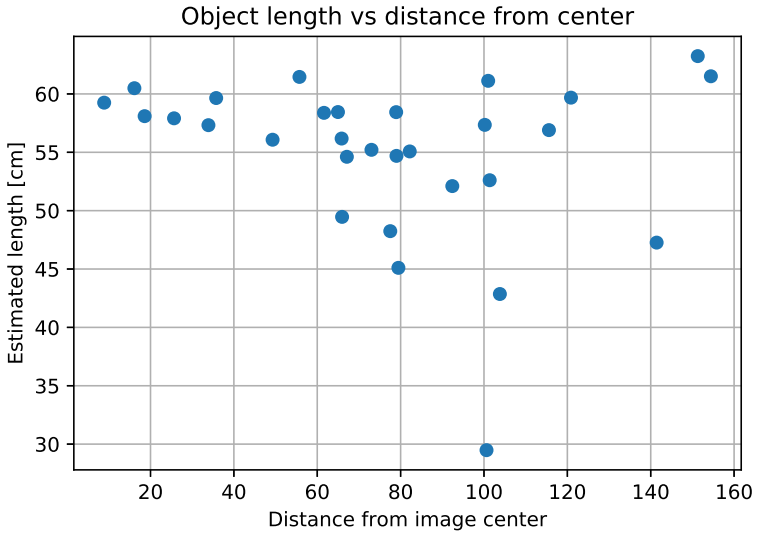


Figure 10.40: 30 length estimates for the same fish, measured at irregular intervals over a time period of 320 frames. The length was plotted against the fish's absolute distance to the image center. The distance from camera z was calculated as the median of a number of hand-selected points along the fish, and the fish was assumed to be perpendicular to the optical axis (i.e. $z_1 = z_2 = z$). The estimated length varies due to contortions of the fish, but the maximum estimated length for a given distance interval is relatively consistent, regardless of the distance to the image center.

Next, the effect of errors in selecting the end points of the fish, and determining their distances to the camera, was considered. These errors may stem from errors in the segmentation, errors in the ellipse fitting algorithm or noise in the depth map. A virtual fish, as illustrated in Figure 10.41 was defined by two coordinates in an image along with their distances to the camera: (x_1, y_1, z_1) and (x_2, y_2, z_2). Initially, both points were placed at the same distance to the camera $z_1 = z_2 = z$, meaning the fish is assumed to be perfectly perpendicular to the optical axis.

The fish's distance from the camera, z, is generally not constant along the fish. This could be due to the fish's position and orientation with regards to the camera, possible segmentation errors resulting in the algorithm selecting the distance to the background instead of the fish, or inaccuracies in the depth map. The effect of small perturbations in z on the length estimate was investigated. Figure 10.42 shows that the absolute percentage error in the calculated fish length is about 1% per cm change in distance from the camera z for objects 100 cm from the camera. The error per cm change in distance decreases for objects further way from the camera.

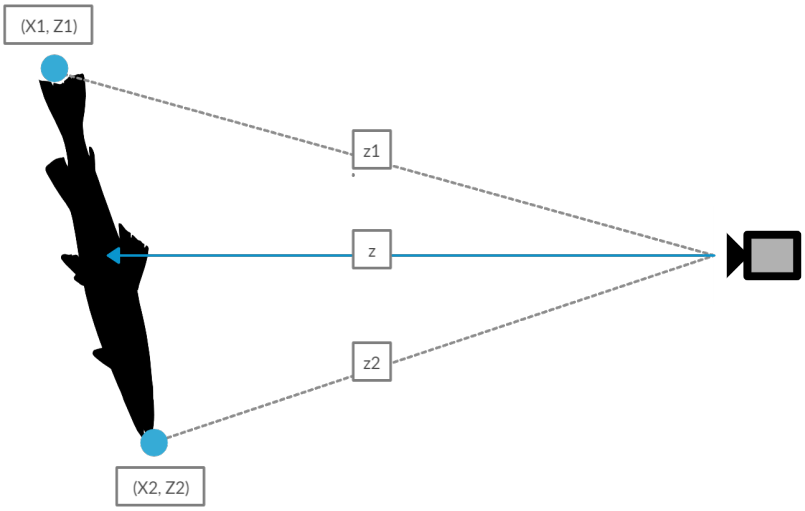


Figure 10.41: 2D illustration of the virtual fish defined by two points marked in blue. Real world coordinates (X_1, Z_1) and (X_2, Z_2) can be calculated from the point's horizontal pixel position in the image x_1 and x_2 and their distances from the camera z_1 and z_2.

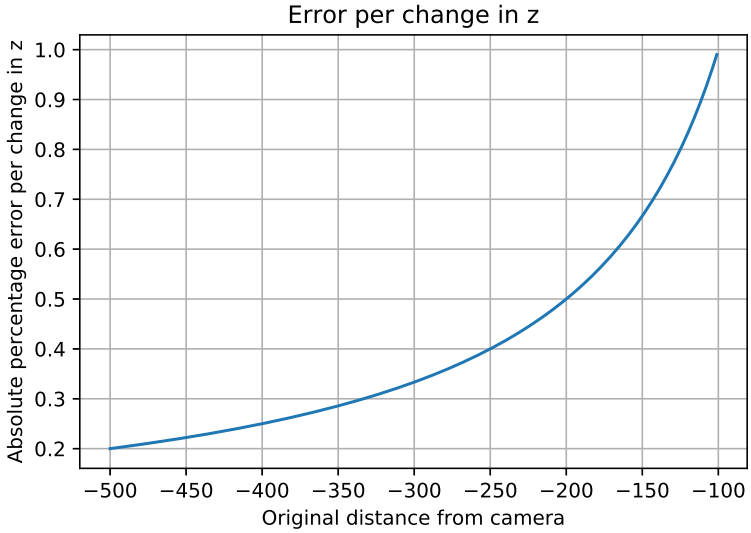


Figure 10.42: A virtual fish with a constant length of 100 pixels was placed in the image at different distances from the camera z. The fish length was estimated at each z and compared the length calculated at $z + 1$. For a fish at 100 cm from the camera, the small perturbation in z changes the length by about 1%. At greater distances from the camera, small changes in z have a much smaller effect on the length estimate, due to the pixel length not changing.

Due to segmentation errors or inaccuracies in the edge point retrieval algorithm, there are likely errors in the placement of the end points as well; changing the pixel length of the fish. Using the virtual fish, with

$z_1 = z_2 = z$, the length was calculated for a left point at $x_2 + i$, $i \in [-30, 30]$ and compared to the original length. The absolute percentage error in the estimated length per pixel change in one point was between 0.98% and 1.1%, regardless of z, as illustrated in Figure 10.43. A change in pixel length has the largest impact near the edges of the image.

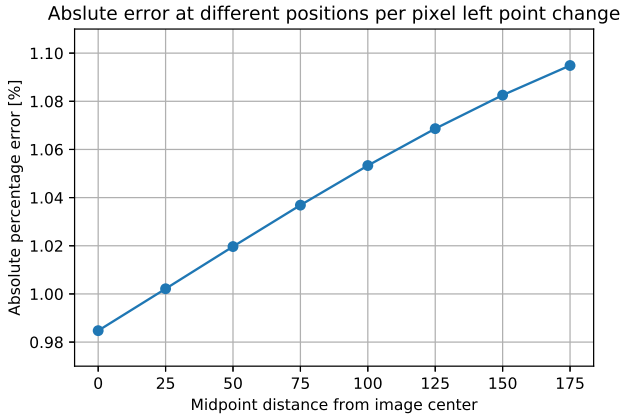


Figure 10.43: A virtual fish of 100 pixels was placed in the image and the absolute percentage error per pixel change in the x-direction for the left point was calculated for various mid-point positions of the object. The error is slightly larger near the image border.

So far, the assumption has been that the fish is perpendicular to the optical axis; meaning that all points on the fish are at the same distance to the camera z. This assumption is of course not generally true. By keeping $z_1 = z$ constant and gradually changing z_2, as illustrated in the simplified 2D diagram in Figure 10.44, the possible resulting error due to this assumption was investigated.

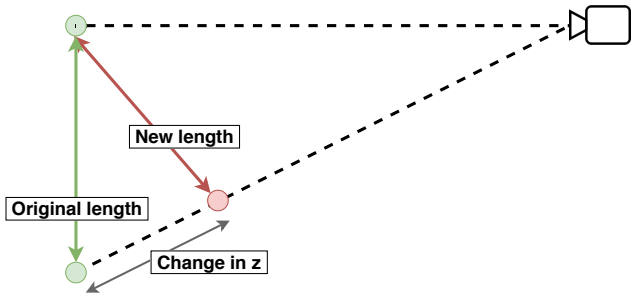
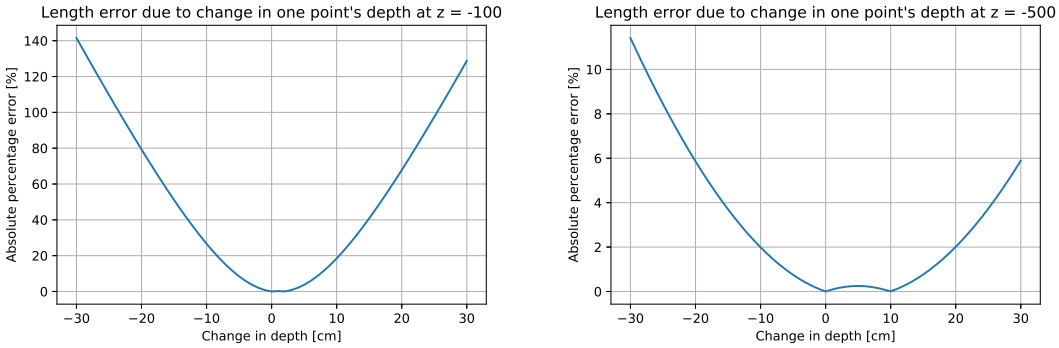


Figure 10.44: A simplified 2D illustration of the virtual fish defined by two points. Only the x-axis (distance from dotted center line) and the z-axis (point distance from camera) is shown. One point's position is kept constant, while the depth of the other is changed. The new estimated length (red line) is compared to the original length (green line).



(a) 100 cm from the camera

(b) 500 cm from the camera

Figure 10.45: Absolute percentage error due to a perturbation in the distance to one of the points defining the virtual fish. It is unsurprising that changing z_2 has a larger influence on the estimated length for objects close to the camera, compared to an object further away; after all both objects have the same pixel length.

Figure 10.45 shows that the difference in estimated length after changing the distance to one point may be quite dramatic, especially for objects close to the camera. Fortunately, the distance estimates are more accurate the closer an object is to the camera. The "bump" between 0 and 10 cm in Figure 10.45 (b) is explained by Figure 10.44; an object's estimated length is minimal when the line between the points is perpendicular to the distance vector, before increasing again.

10.3.2 Estimating distance to camera

If the segmentation is fairly accurate and the fish is well positioned, the length estimate should be consistent across frames for the same fish. The pixel coordinates of the fish's head and tail, (x_1, y_2) and (x_2, y_2) need to be determined as well as their distances from the camera z_1 and z_2.

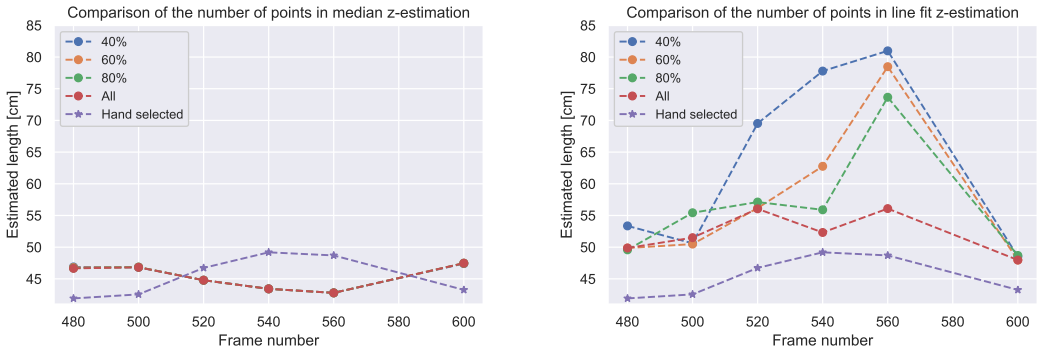
For the sake of simplicity, the fish is assumed to always be straight. Following a straight line along the major axis of the fitted ellipse, the points are placed at the intersections of the line and the segmentation border. In the case of more than two intersections, the two intersections furthest from the fish's midpoint are selected.

Due to uncertainties in the pixel points and inherent inaccuracies in the depth map, selecting the distances z_1 and z_2 at the pixel points would in most cases return the distance to the background rather than the fish. Instead, the distances were estimated using various methods which were later compared. Initially, $z_1 = z_2$ were set equal to the median of a percentage of the distances, nearest to the midpoint, along the major axis of the fish. Taking the median removes outliers, and using distances near the midpoint lowers the likelihood that distances to points outside the fish are included.

Two attempts were also made at fitting a function to a selection of distances along the fish in order to estimate z at the end points. A plane

was fitted to all segmented points on the fish, and a line was fitted to points along the major axis. All length estimates from these functions were compared to lengths calculated using hand-selected end points and the observed distances at these points.

First, the percentage of points used in the line fit and the median distance estimates was evaluated and compared to the hand-selected point estimates. Figure 10.46 shows that the median estimate hardly changed as the percentage of points used in the estimate grew from 40% to all. However, the line fit estimates grew closer the the hand-selected point estimates as more points were included.



(a) Distance estimated using the median

(b) Distance estimated using line fit

Figure 10.46: The distances z_1 and z_2 are estimated using the median and a line fitted to $k = \{40, 60, 80, 100\}$% of the distance measurements along the major axis of the fitted ellipse. k% of the closest points to the ellipse midpoint are used in each length estimate. When using the median to estimate distance, the length estimate hardly differs for different k. Increasing the number of points used in the line fit seems to return a more stable length estimate, increasingly closer to the estimate based on hand-selected points.

Figure 10.47 shows estimated lengths using the various methods (median, line fit and plane fit) to estimate z_1 and z_2 over time on two hand-segmented fish. The frames were chosen so that the fish were relatively well-positioned in each frame. One frame was selected to illustrate how the methods estimate the distances differently (see Figure 10.48).

Overall, using the median to determine the distance z seems to follow the hand-selected points most closely. It is also the most consistent method. While the lengths of the fish are known (71 cm, 54 cm and 41 cm), they are not marked, so we do not necessarily know the true length of the fish used to test the distance estimation methods. However, the length estimated in the right graph in Figure 10.47 is slightly concerning, as it is not clear whether the lengths in the graph are due to the length of the 41 cm long fish being overestimated by the fitted line and plane, or the length of the 54 cm long fish being generally underestimated when using the median or the hand-selected points.

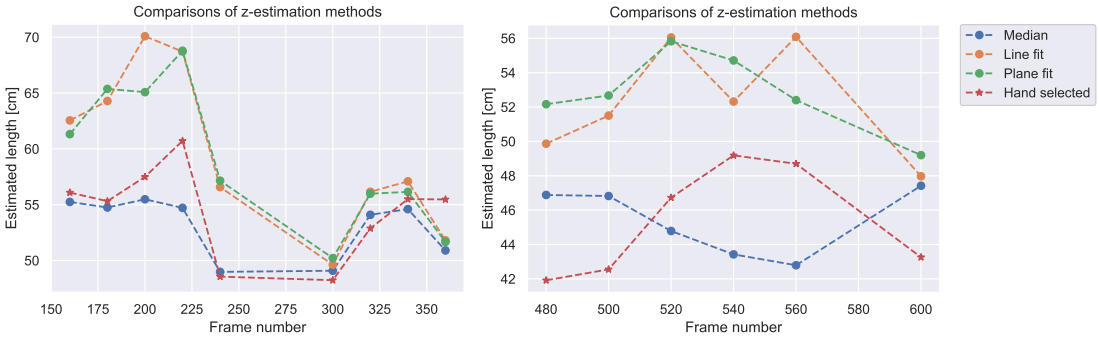


Figure 10.47: Estimation of the length of a hand-segmented fish over time. It seems that estimating the distance z using the median of points along the major axis of the fish is a relatively stable method with results relatively close to the hand-selected points. The line fit and the plane fit methods seem to consistently overestimate the length compared to the hand-selected points.

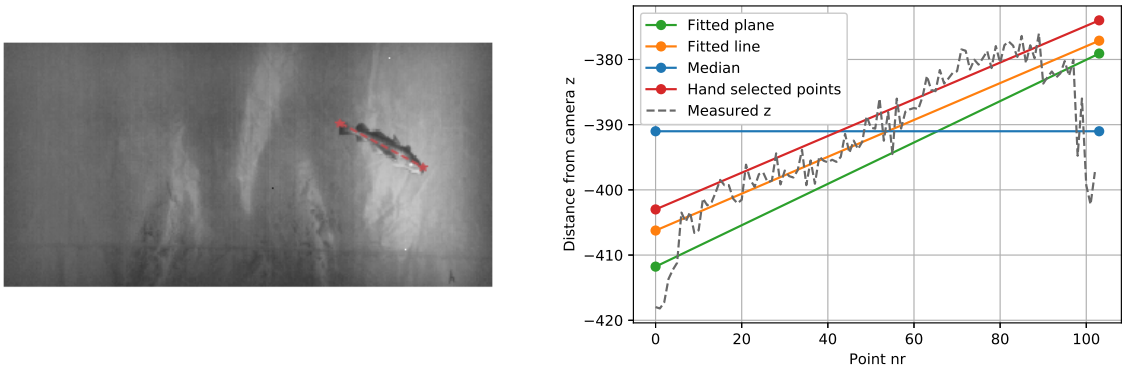
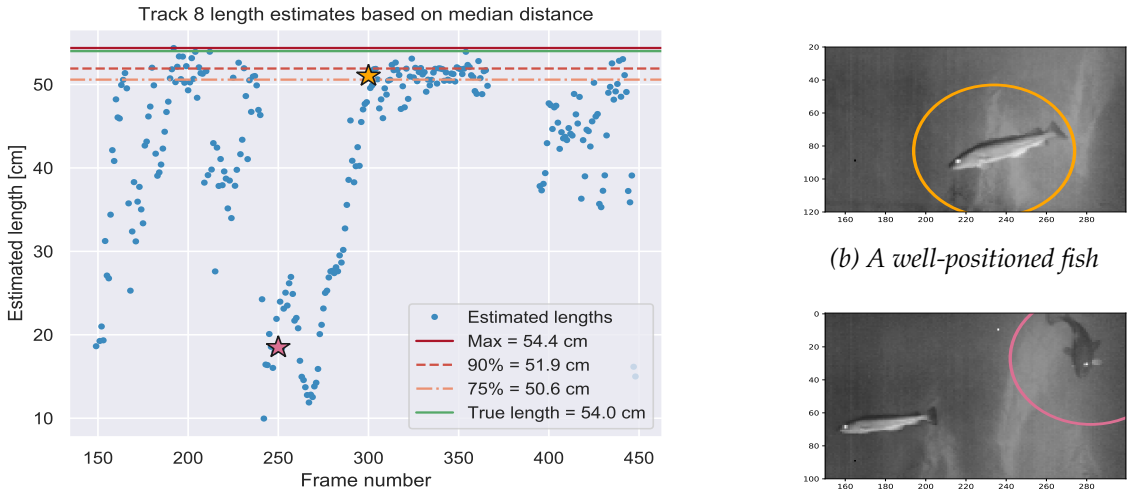


Figure 10.48: Frame 540 illustrates how the different methods return different length estimates. The median returned the lowest length estimate in this frame, likely due to the increase in z along the major axis. The line fit, plane fit and hand-selected points have similar angles, but the hand-selected distances are clearly closer than the plane fit distances (the furthest away from the camera and returning the largest estimated length).

10.3.3 Length estimation based on multiple observations

Using the tracking validation set, the fish are followed over longer periods, allowing for the length to be estimated multiple times. Additionally, all three fish are present in certain frames, making it possible to determine which track belongs to which fish. Taking all tracks that contain more than one observation, final length estimates were produced for each track based on length at the 75th percentile, the 90th percentile and the maximum of all estimated lengths in that track. Figure 10.49 illustrates how the final length for the track with the most observations may be calculated. An example of a well-positioned fish, resulting in a good length estimate, and an example



(a) Length estimates over time

(b) A well-positioned fish

(c) A badly positioned fish

Figure 10.49: (a) shows the estimated length of track 8, with z estimated as the median of all distances along the major axis, over a period of more than 400 frames. The true length of the fish, the maximum length estimate and the length at the 90th and 75th percentile are shown in the plot; in this case the maximum length estimate is the closest to the true length. Two length estimates are marked with stars in the plots, (b) and (c) show the position of the fish at these points.

of a badly positioned fish resulting in a poor length estimate are marked in this figure.

Selecting the tracks with the most observations for each fish, the length was calculated for every observation in that track using median, line fit and plane fit to estimate the distance z. Figure 10.50 illustrates the 75th and 90th percentile as well as the maximum of all estimated lengths for each method. No method correctly estimated the length for all three fish. The length of the largest fish was generally underestimated, while the length of the smallest fish was always overestimated. The maximum estimated length value using the median distance resulted in the smallest error for the three tracks: the average absolute percentage error was 10.8%.

That being said, I am hesitant to conclude that this method is better than the others. It overestimates the smallest fish by more than 17% of the original fish length, and underestimates the largest fish by 14%. Using the maximum length estimated may also make this method sensitive to outliers. If all tracks in the test set with more than one observation were included, taking the 90th percentile of lengths estimated based on a fitted line returned the lowest average absolute percentage error: 9.6%. 6 tracks were included in this calculation; the largest fish was included thrice as it entered and exited the frame multiple times, the second-largest fish was included twice and the smallest fish was included only once. The maximum of length estimations based on a median distance returned an average absolute percentage error of 10.8% on this dataset as well. I assume

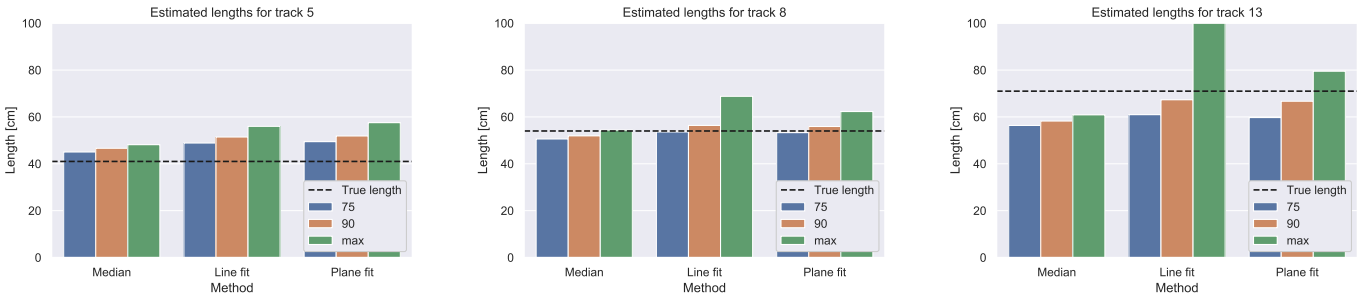


Figure 10.50: Three tracks, each belonging to a different fish were selected. The length was calculated for every observation, using the median distance, and distance estimated using a fitted line and a fitted plane. A final length estimate was calculated as the 75th percentile, the 90th percentile and the maximum estimated length.

that if the calculations were repeated on a different dataset with multiple larger fish, the line fit or plane fit methods would return the best results as they generally return larger estimates.

It should be noted, once again, that the orientation of the fish very heavily influences the result of the length estimation. In the tracking validation dataset, the largest fish appeared in three separate tracks. Its length was generally underestimated in track 13 illustrated in Figure 10.50, but in an earlier (and shorter) track, its orientation with regards to the camera seems to have been optimal for length estimation; the maximum of the line fit lengths and the plane fit lengths both resulted in a length estimate which deviated by less than 2% from the true length.

10.3.4 Length estimation summary

After noise removal, adaptive Gaussian thresholding returned an accuracy of a little less than 60% on the segmentation test set. Some of the fish are near perfectly segmented, but it is likely that the extreme parts of the fish will be missing in many frames. A one pixel change in pixel length resulted in the length estimate changing by around 1%. One might expect the fish length to be continuously underestimated. However, the length of the smallest fish in the test set was continuously overestimated, regardless of the method used to estimate its distance to the camera z.

Depending on the fish's distance to the camera, changing the distance of one of the end points used to measure the fish length by 1 cm could result in a 2% change in the fish length. Due to noise in the depth map and segmentation errors, errors in z are likely to affect the length estimate. In an attempt to alleviate errors in z, attempts were made at estimating the distance by fitting a line or a plane to selected points along the fish, or simply taking the median of points along its major axis. The length is estimated once for every observation in a track. The final length estimate for each track must be based on a portion of the largest lengths in that track. Results on the 3 Cod dataset indicate that using line fitting to estimate

distance, and then estimating the final length based on the 90th percentile of the lengths in the track, produce the best results. Using this approach to estimate the lengths of fish in six tracks resulted in an average absolute error of 9.6%.

However, this method still consistently underestimates small fish and overestimates larger fish. I believe more data with a range of known fish lengths is needed in order to truly analyse the issues with the length estimation and develop a method which more consistently returns length estimates with less deviation from the true lengths.

10.4 Results on the Fish Schools Dataset

As seen in chapter 4, the depth maps in this dataset contain a much larger amount of None-entries compared to the 3 Cod dataset. Almost 30% of the total depth map entries in this dataset are None-entries. While the handling of None-entries in the 3 Cod dataset hardly affected the final results, it now has a major effect. Replacing these values with 0 is not a viable option, as the thresholding methods are greatly affected by this and end up mostly segmenting the None-entries as illustrated in Figure 10.51.

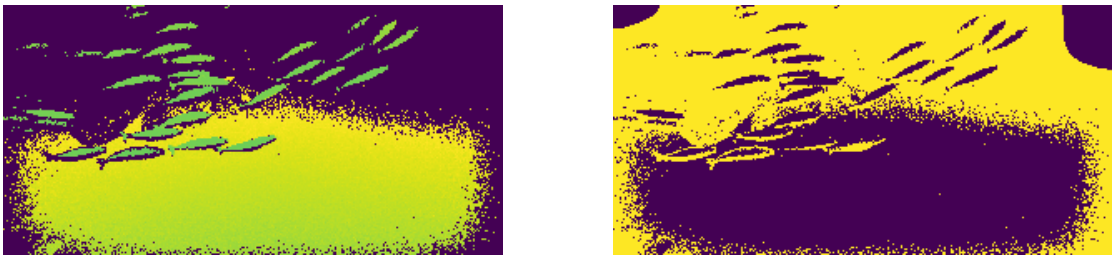
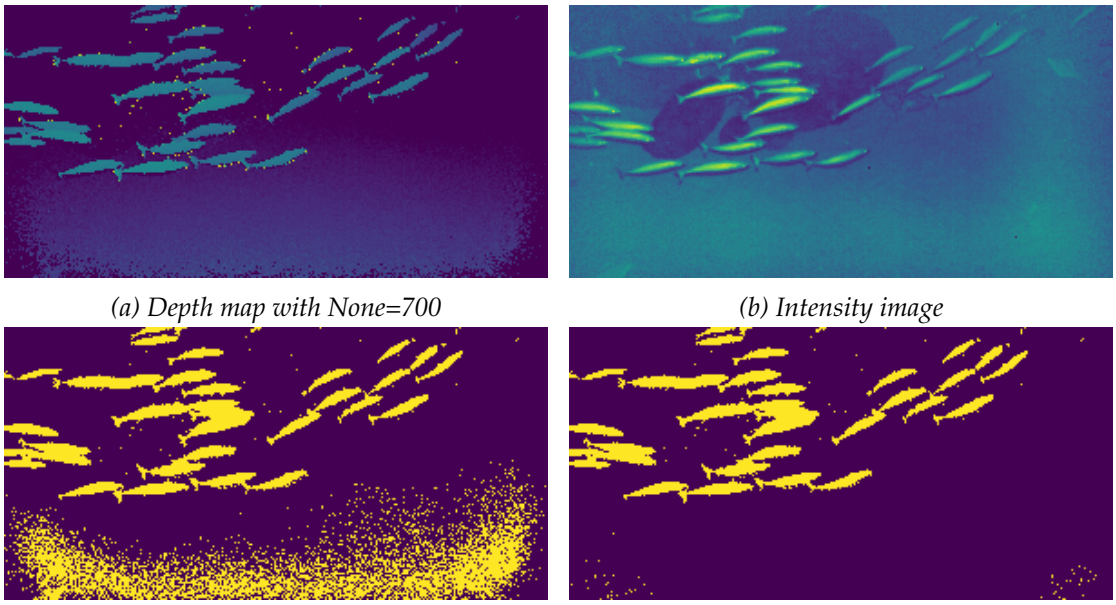


Figure 10.51: Left illustrates the depth map with None-entries replaced by 0. Right is the resulting segmentation using adaptive Gaussian thresholding with block size = 111 and C = 3. All None-entries are perceived to be close to the camera.

Instead, every None-entry is considered to be as far away from the camera as possible (i.e. at a depth of 700 cm). In addition, the constant C is increased at the risk of missing some fish near the bottom of the image where the background depth is calculated, but with the benefit of not segmenting large parts of the background. Figure 10.52 shows how this improved the segmentation. Replacing None-entries with the maximum depth value achieved better results than an alternative algorithm which ignored None-entries and calculated the threshold based on only entries with a high-certainty value, as the alternative algorithm weighted high-certainty entries in areas with many None-entries too heavily.

Another issue caused by the large amount of None-entries is the Kalman filter being initialized with initial distance to the camera $z_0 = \texttt{None}$. While the Kalman filter is able to return predictions when it is updated with None-entries, it cannot be initialized with None-entries. By letting z_0 be the median of distances along the fish's centerline, rather than distance at the



(a) Depth map with None=700

(b) Intensity image

(c) C = 3

(d) C = 20

Figure 10.52: Using a C-value which resulted in a relatively successful segmentation on the 3 Cod dataset results in large parts of the floor being segmented in this case. An increased $C = 20$ removes most of the false positives while still segmenting the fish.

the fish's mid point, the Kalman filter ran without issue on all Fish Schools tracklets tested.

Most parts of this dataset contain too many fish to run the tracklet connector. As the number of tracklets increases, so does the complexity of the assignment problem. A large number of fish swimming close to each other means that every merge situation has multiple solutions in the tracklet connector. The fish having approximately the same size and swimming in the same general direction with the same speed make it more unlikely that the correct solution is returned. The tracklet linker was run on certain parts of the Fish Schools dataset with smaller numbers of fish. Unsurprisingly, it struggled with resolving merges and ultimately returned very little additional information compared to the information from the tracklet generator.

Ten frames were selected from the dataset. The fish were manually counted and the count was compared to estimated number of fish using the tracklet generator and the tracklet linker. The tracklets were generated and connected using the cost functions that performed the best on the 3 Cod dataset (i.e. using distance with σ_{dist} = 15 to generate tracklets, and distance and velocity angle with σ_{dist} = 15 and σ_{angle} = 25 to connect them).

Figure 10.53 indicates that the algorithm is able to keep track of at least ten fish reasonably well, but when the number of fish rises above thirty, the number of occlusions is too large for the algorithm to handle.

Nevertheless, the tracklet generator returns information with regards

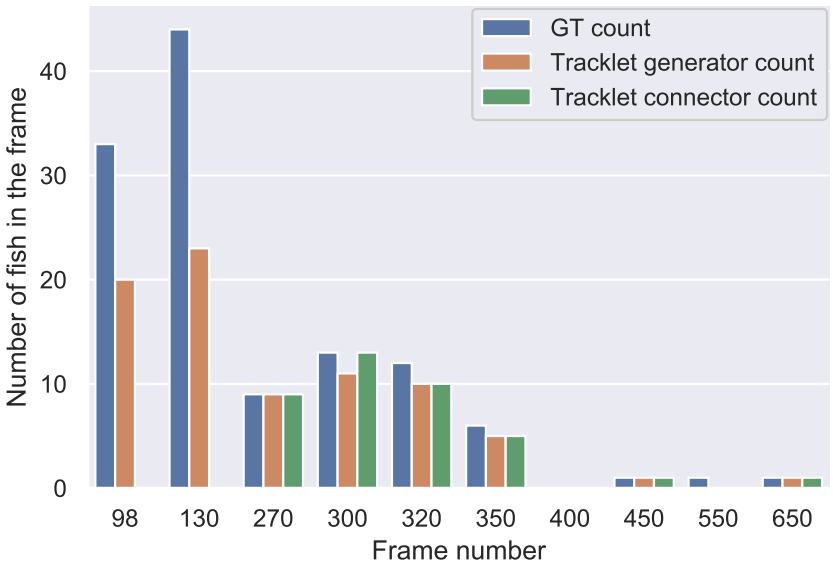


Figure 10.53: The bar plot illustrates the true number of fish in each test frame, along with the estimated number of fish using only the tracklet generator (equal to number of connected components after segmentation) and the count after applying the tracklet linker as well. The tracklet linker failed to run on the two first example frames due to the large number of fish. Otherwise it produced similar results to the tracklet generator.

to the direction and speed of the school of fish. By estimating the length of multiple tracklets, a length estimate for an "average" fish in the school might be estimated. An attempt was made at estimating the lengths of the fish observed between frame 200 and 400 in this dataset. The length for each observation was estimated with a fitted line to determine the distance at the end points, and the final length estimate for each track with more than ten observations was based on the 90th percentile of the lengths in the track. The average length of the fish was estimated to be about 43 cm, which is plausible. The maximum length estimated was almost 75 cm; in this case, the tracking algorithm failed to recognize two fish merging. Disregarding this outlier, the algorithm returned 19 length estimates between 24.8 cm and 59.6 cm.

Chapter 11

Final algorithm overview

The proposed pipeline was presented in Chapter 5. After having investigated the performance of various segmentation, tracking and length estimation methods, the final algorithm is presented in this chapter with more details.

The final algorithm broadly consists of three parts. The first part is a tracklet generator, illustrated in Figure 11.1, which detects fish in every frame and attempts to link them to observations in the previous frame. The second part is a tracklet linker, illustrated in Figure 11.2, which applies a Kalman filter to all tracklets from the tracklet generator and attempts to link the tracklets to form the final tracks. The tracklet linker works iteratively to solve occlusions by identifying and copying tracklets that likely contain observations with more than one fish. Finally, a length estimation algorithm can produce the final length estimate for each fish based on the tracks.

Due to the variability in intensity of the fish and the background, segmentation should be done using the depth map, or alternatively combining results from the depth map and the intensity image. Due to a gradient in the depth map, adaptive Gaussian thresholding with noise removal using morphological operations and segmentation using background subtraction returned the best validation metrics results. Of the two, adaptive Gaussian was used in the final algorithm for reasons of computational speed. The success of these methods depends on a fairly homogeneous background. All foreground objects are assumed to be fish, which may not always be the case. If more accurate segmentation is needed, for example for classification purposes, background subtraction may be applicable, perhaps only for certain frames. Both methods occasionally fail to detect fish, usually near the edges of the image or on occasions where the fish is especially far from the camera.

The simple tracklet generator algorithm based on distance between observations worked well on both the 3 Cod dataset and the Fish Schools dataset. The tracklet linking algorithm successfully identified and solved merge situations and occurrences of lost detections in the 3 Cod dataset. It was determined that the best cost function combined distance and direction comparisons. The distance and the direction was determined

using Kalman filters, providing velocity vector estimates and allowing for estimating the position of a fish from earlier observations.

Investigating the effect of errors in the placement of the end points for length estimation revealed that small perturbations in the pixel position or distance to the camera at these end points had the potential to drastically change the length estimate. Due to inherent inaccuracies in the depth map, especially near the edges of the fish, and errors in the segmentation, end point errors are likely to affect the length estimates. Using the distance measured at the end points proved to be too imprecise; instead attempts were made at estimating the distance using the median of distances along the center line of the fish, a line fitted to the center line of the fish and a plane fitted to all points of the fish.

The final length was estimated as 90th percentile of the lengths estimated for every observation (using the binary mask from the segmentation, the fitted ellipse modelling the fish and a fitted line to estimate the fish's distance to the camera). On the test set with 501 frames and known fish lengths, this method achieved an average absolute percentage error of 9.6%. Due to the small sample size, the error in the length estimate was difficult to ascertain. A dataset with more fish with known lengths is necessary for an in-depth study of length estimation. It is, however, clear that a large number of estimates per fish is an advantage; the fish are relatively seldom in an optimal position for length estimation.

The tracklet linking algorithm was not successful on the Fish Schools dataset, indicating that there exists an upper limit for the number of fish the algorithm can track. A tentative threshold of ten fish is suggested, but this is highly dependent on the quality of the input data and the behaviour of the fish (a school of fish swimming close to each other in the same direction is suboptimal for this type of algorithm).

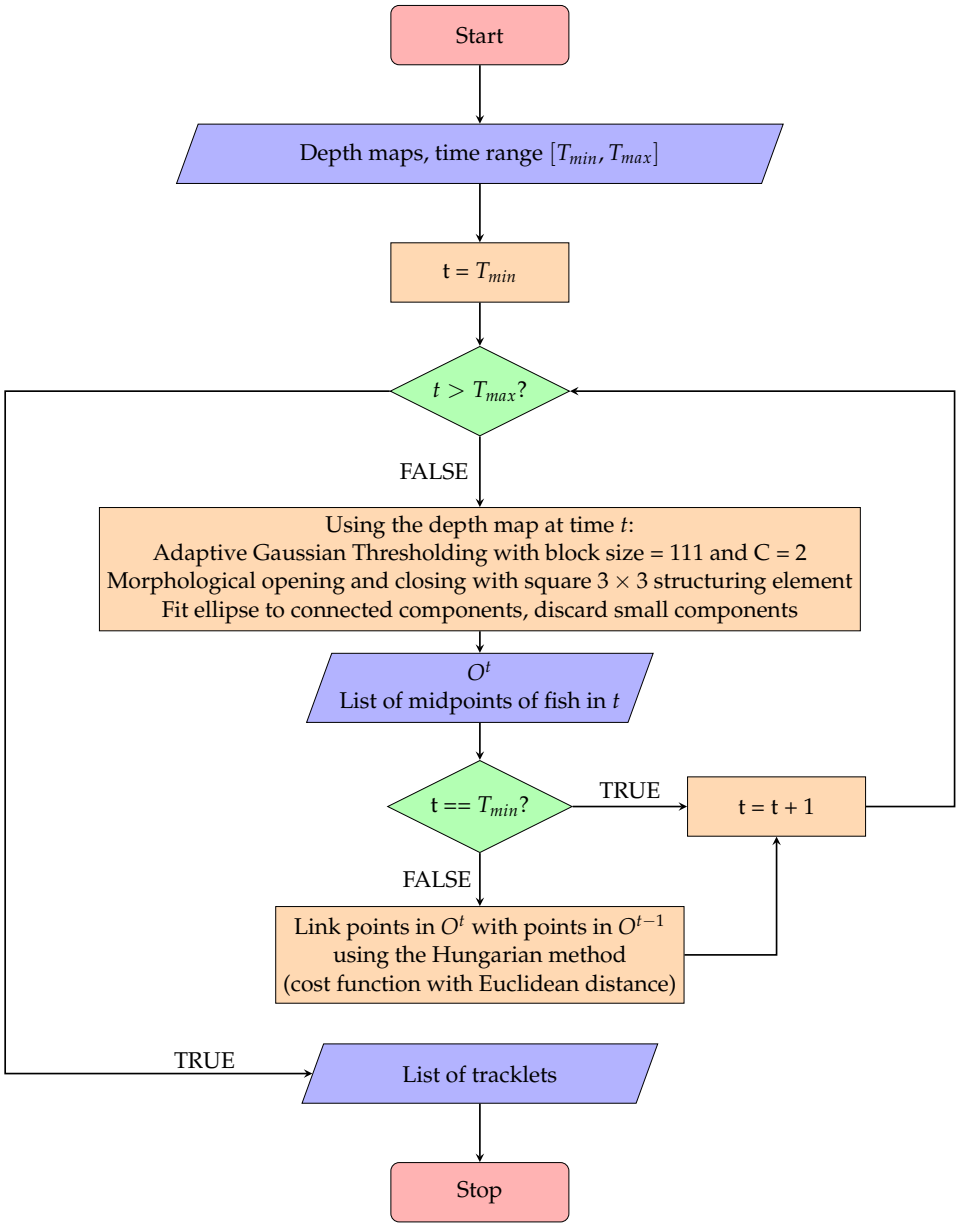


Figure 11.1: Tracklet generator algorithm overview.

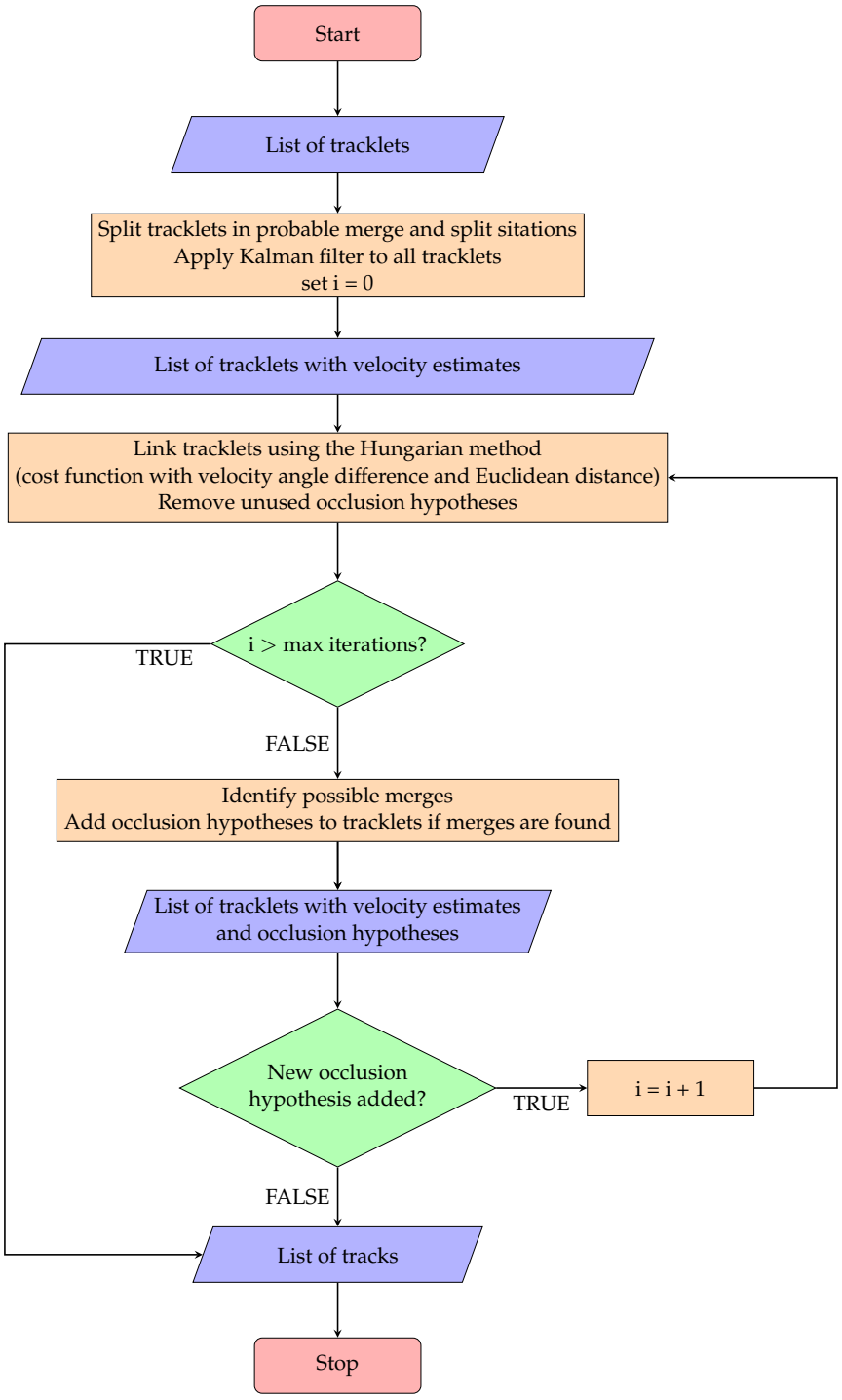


Figure 11.2: Tracklet linker algorithm overview. Note that occlusion hypothesis (copies of tracklets that likely contain observations with more than one fish) are removed if they are unused, i.e. linked to the source and sink in the proposed solution. This is in order to prevent the model from adding false tracks.

Part IV

Conclusion and Future Work

Chapter 12

Conclusion

12.1 Conclusion

The aim of this thesis was to detect, track and estimate the length of free-swimming fish captured by an underwater 3D camera. This study is largely based on a video containing three codfish filmed in a pool. The detection of the fish was based on Gaussian adaptive thresholding, though iterative background subtraction also returned promising results. The tracking algorithm was based on solving a minimum-cost flow problem multiple times. Initially, observations were linked frame-by-frame to form tracklets. Using information about the entire time sequence, occlusions were identified and tracklets were linked to form continuous tracks. Two cost functions were developed and tested for the tracklet generation and the tracklet linking steps. The cost functions were based on the fish's position and velocity rather than intensity or texture due to the variability in illumination and the fish being the same species with no obvious markings separating them. The length was estimated for every observation. Knowing that the contortions of the fish often results in the length being underestimated, the final length estimate should be based on a high percentile of all the length estimates in a track.

The segmentation and tracking were evaluated separately. The segmentation was evaluated using four validation metrics. Lowering the number of false positives was a major concern and deemed more important than a perfect segmentation of the edges of the fish in every frame. Gaussian adaptive thresholding followed by noise removal with morphological operations achieved a 59% median accuracy, 99% median mean error, a median boundary F1 score of 0.62 and a median Jaccard index of 0.53, calculated using 93 hand-segmented images (including some devoid of fish).

The final tracking algorithm, using a cost function based on Euclidean distance to generate tracklets and a cost function combining Euclidean distance and velocity angle difference to link tracklets, achieved a multiple object tracking accuracy (MOTA) of 92%. All fish in the 501-frame tracking validation data set retained their original ID through occlusions. The multiple object tracking precision was 3.4 , meaning the midpoint of each

observation was on average 3.4 pixels removed from the hand-selected midpoint.

The estimated lengths for each observation were computed based the length of the line segment between two end points of the fish. The distance to the camera of these two points proved difficult to determine, as the uncertainty of the depth estimates increases near the edges of the fish. Therefore, the distances at the end points were estimated by fitting a line to all distances measured along the major axis of the fish. The final length for each track was estimated as the 90th percentile of all lengths calculated in that track. This method achieved an absolute percentage error of 9.6%, based on six long tracks in the test set. These are promising results, but more data is needed for a true analysis of the error in the length estimation. The estimated error was based on only three data points: three codfish with known lengths, some repeatedly entering and exiting the frame and thus creating more tracks.

Applying the algorithm to a different dataset containing several schools of fish revealed the limitations of this algorithm. One is the fact that the segmentation parameters currently needs to be determined for each dataset. Another is that an increasing number of fish decreases the efficiency and accuracy of the tracking. The tracklet linking algorithm was not able to solve the occlusions in this dataset due to the fish swimming closely together and often in the same direction. That being said, the adaptive Gaussian threshold produced promising results with minor parameter changes and the tracklets generated could potentially be used to estimate the average length of the fish in the frame.

12.2 Future work

This thesis provides a basis for more complex systems with more extensive functionalities useful for fishing, fish farming and conservation. I have outlined some suggestions below for improvements and further development of the algorithms presented in this thesis.

- **Biomass estimation**

  Provided that the tracking and length estimations for each fish are accurate, the biomass of the fish observed in the frame can be calculated. Many studies have been done on the relationship between the length of a fish and its weight. Fish length and weight relationships are generally calculated: [62]

  $$W = aL^b \tag{12.1}$$

  where W and L represent the fish's weight and length respectively, and a and b are constants estimated using empirical evidence for each fish species. The regression coefficient b is expected to be roughly equal to 3 for all fish species, as the volume of a 3-dimensional object is roughly L^3, where L is the length of the object. [6] For north sea cod,

$a = 0.0081$ and $b = 3.0502$, while the coefficients for atlantic salmon are $a = 0.0274$ and $b = 2.7802$. [62]

- **Species classification**

  Accurate biomass calculations are evidently only possible if the fish species are known. The segmentation and tracking provide a basis for a classification algorithm able to determine the species of the fish. The initial segmentation indicates the locations of possible fish; a second local segmentation focusing only on these areas may help to restore details lost in the morphological noise removal process. Using the distance data available for each observation, along with the calculated velocity vector and the positions of potential other fish in the frame, ideal observations can be selected and analysed. These observations should be as close to the camera as possible and with a velocity vector perpendicular to the optical axis so that characteristic features such as the fins are likely to be well segmented.

  Previous studies have used shape and texture analysis to determine the fish species. Fabic et al. [14] used Zernike moments to match segmented fish to moment signatures of two fish species stored in a database. While the system did not quite match the human count, it was deemed "a viable alternative to manual methods." Spampinato et al. [65] combined texture features and shape features extracted using the Curvature Scale Space transform and the histogram of Fourier descriptors of boundaries to classify classify ten different fish species with an average correct rate of 92%. Larsen et al. [30] analysed shape and texture of three fish species and achieved a 76% recognition rate (on a manually outlined training set). Lee et al. [32] used a hierarchical classification tree and a set of color, shape and texture properties to classify ten fish species. They achieved an average recall rate of 90%.

  In recent years, deep learning approaches have become more popular. Salman et al. [58] achieved a correct classification rate of more than 90% using a convolution neural network (CNN) model to extract species-dependent features of fish. Rathi et al. [52] also used image processing techniques to enhance images and a CNN to classify 23 fish species with an accuracy of 97%. Allken et al. [2] developed a deep learning neural network specifically for trawl monitoring with an accuracy of 94% for three species of fish. Deep learning-based models in general seems to achieve a higher accuracy when compared to other models, but the drawback is the amount of data required. The development of the model in [2], for example, was based on a thousand manually curated images, from which a larger synthetic dataset was produced.

- **Improving the detection and tracking**

  While the tracklet generator described in this thesis is able to return an estimate of the number of fish in each frame, the accuracy

decreases with an increasing number of fish. Deep learning can also be used to detect the fish and to aid in the tracking. While local thresholding methods performed well on the 3 Cod dataset, a deep learning based detector may be able to distinguish individual fish in situations where they are detected as one by the segmentation methods presented here. Xiuli et al. [35] used a so-called Fast R-CNN (region-based convolutional neural network) to simultaneously detect and classify free-swimming fish, achieving a mean average precision of 81%. Wang et al. [71] used a scale-space Determinant of Hessian blob detector to detect fish heads and a specifically adapted CNN to identify the head pattern of each fish in the frame for data association across frames.

In this thesis, I have outlined the shortcomings of some of the features used in the data association step. A CNN such as the one described in [35] may be used to improve the cost function. The success of the deep learning approaches depends largely on the amount, and quality, of the labelled training data available. Synthetic data can be produced to expand smaller datasets.

- **Thorough error evaluation of the length estimation**

  Finally, more samples are needed to perform a proper analysis of the length estimation. The reasons why the error is generally larger for the smallest and largest fish, while the length of the medium-sized fish tends to be almost perfectly estimated, need to be examined using a larger dataset. It is possible that some of the methods tested in this thesis under- or overestimates the length in a consistent and predictable manner that is not apparent in this dataset.

Bibliography

[1] Fritz Albregtsen. 'Non-Parametric Histogram Thresholding Methods - Error Versus Relative Object Area'. en. In: *Proc. 8th Scadinavian Conf. Image Analysis* (1993), pp. 273–280.

[2] Vaneeda Allken et al. 'Fish Species Identification Using a Convolutional Neural Network Trained on Synthetic Data'. en. In: *ICES Journal of Marine Science* 76.1 (Jan. 2019). Ed. by Richard O'Driscoll, pp. 342–349. DOI: 10.1093/icesjms/fsy147.

[3] Vicente Atienza-Vanacloig et al. 'Vision-Based Discrimination of Tuna Individuals in Grow-out Cages through a Fish Bending Model'. In: *Computers and Electronics in Agriculture* 130 (Nov. 2016), pp. 142–150. DOI: 10.1016/j.compag.2016.10.009.

[4] Bilal Bataineh et al. 'Adaptive Thresholding Methods for Documents Image Binarization'. In: *Pattern Recognition*. Ed. by José Francisco Martínez-Trinidad et al. Vol. 6718. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 230–239. DOI: 10.1007/978-3-642-21587-2_25.

[5] Keni Bernardin, Alexander Elbs and Rainer Stiefelhagen. *Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment*. en. 2006.

[6] Jon Kenton Tarsus Brodziak. *Fitting Length-Weight Relationships with Linear Regression Using the Log-Transformed Allometric Model with Bias-Correction*. Tech. rep. 2012.

[7] John Canny. 'A Computational Approach to Edge Detection'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[8] Xi Chen, Xiao Wang and Jianhua Xuan. 'Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques'. In: *arXiv:1802.01235 [cs]* (Feb. 2018). arXiv: 1802.01235 [cs]. URL: http://arxiv.org/abs/1802.01235 (visited on 08/06/2020).

[9] Meng-Che Chuang et al. 'Automatic Fish Segmentation via Double Local Thresholding for Trawl-Based Underwater Camera Systems'. en. In: *2011 18th IEEE International Conference on Image Processing*. Brussels, Belgium: IEEE, Sept. 2011, pp. 3145–3148. DOI: 10.1109/ICIP.2011.6116334.

[10] Scott Cohen. 'Finding Color and Shape Patterns in Images'. PhD thesis. Stanford, CA, United States: Stanford University, May 1999. URL: http://infolab.stanford.edu/pub/cstr/reports/cs/tr/99/1620/CS-TR-99-1620.pdf.

[11] Gabriela Csurka, Diane Larlus and Florent Perronnin. 'What Is a Good Evaluation Measure for Semantic Segmentation?' en. In: *Procedings of the British Machine Vision Conference 2013*. Bristol: British Machine Vision Association, 2013, pp. 32.1–32.11. DOI: 10.5244/C.27.32.

[12] R.W.D. Davies et al. 'Defining and Estimating Global Marine Fisheries Bycatch'. en. In: *Marine Policy* 33.4 (July 2009), pp. 661–672. DOI: 10.1016/j.marpol.2009.01.003.

[13] Tim Dobbert. *Matchmoving: The Invisible Art of Camera Tracking*. San Francisco, Calif. ; London: SYBEX, 2005.

[14] J. N. Fabic et al. 'Fish Population Estimation and Species Classification from Underwater Video Sequences Using Blob Counting and Shape Analysis'. In: *2013 IEEE International Underwater Technology Symposium (UT)*. Mar. 2013, pp. 1–6. DOI: 10.1109/UT.2013.6519876.

[15] FAO, ed. *Contributing to Food Security and Nutrition for All*. en. The State of World Fisheries and Aquaculture 2016. Rome, 2016.

[16] Andrew W. Fitzgibbon and Robert B. Fisher. 'A Buyer's Guide to Conic Fitting'. In: *BMVC*. 1995. DOI: 10.5244/C.9.51.

[17] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Fourth. Pearson, 2018.

[18] Lawrence C Hamilton and Melissa J Butler. 'Outport Adaptations: Social Indicators through Newfoundland's Cod Crisis'. en. In: *Human Ecology Review* 8.2 (2001), p. 11.

[19] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Vol. 1. Addison-Wesley, 1992.

[20] Robert M. Haralick, Stanley R. Sternberg and Xinhua Zhuang. 'Image Analysis Using Mathematical Morphology'. en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.4 (July 1987), pp. 532–550. DOI: 10.1109/TPAMI.1987.4767941.

[21] Joao F. Henriques, Rui Caseiro and Jorge Batista. 'Globally Optimal Solution to Multi-Object Tracking with Merged Measurements'. en. In: *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 2470–2477. DOI: 10.1109/ICCV.2011.6126532.

[22] Morten Hjorth-Jensen. *Data Analysis and Machine Learning: Linear Regression and More Advanced Regression Analysis*. en. Dec. 2019. URL: https://compphysics.github.io/MachineLearning/doc/pub/Regression/pdf/Regression-minted.pdf.

[23] D.P. Huttenlocher, G.A. Klanderman and W.J. Rucklidge. 'Comparing Images Using the Hausdorff Distance'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.9 (Sept./1993), pp. 850–863. DOI: 10.1109/34.232073.

[24] R. E. Kalman. 'A New Approach to Linear Filtering and Prediction Problems'. en. In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. DOI: 10.1115/1.3662552.

[25] Khurram Khurshid et al. 'Comparison of Niblack Inspired Binarization Methods for Ancient Documents'. In: *IS&T/SPIE Electronic Imaging*. Ed. by Kathrin Berkner and Laurence Likforman-Sulem. San Jose, CA, Jan. 2009, 72470U. DOI: 10.1117/12.805827.

[26] Youngjoo Kim and Hyochoong Bang. *Introduction to Kalman Filter and Its Applications | IntechOpen*. July 2018. URL: https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications (visited on 16/02/2020).

[27] J. Kittler and J. Illingworth. 'Minimum Error Thresholding'. In: *Pattern Recognition* 19.1 (1986), pp. 41–47.

[28] H. W. Kuhn. 'The Hungarian Method for the Assignment Problem'. en. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. DOI: 10.1002/nav.3800020109.

[29] Mohamed Laaraiedh. 'Implementation of Kalman Filter with Python Language'. en. In: *The Python Papers Journal* (June 2009), p. 5.

[30] Rasmus Larsen, Hildur Olafsdottir and Bjarne Kjær Ersbøll. 'Shape and Texture Based Classification of Fish Species'. en. In: *Image Analysis*. Ed. by Arnt-Børre Salberg, Jon Yngve Hardeberg and Robert Jenssen. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 745–749.

[31] H. Lee and R. Park. 'Comments on "An Optimal Multiple Threshold Scheme for Image Segmentation'. In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.3 (May 1990), pp. 741–742. DOI: 10.1109/21.57290.

[32] Kyoung Mu Lee and ACCV, eds. *Computer Vision - ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5 - 9, 2012; Revised Selected Papers. Pt. 1: ...* eng. Lecture Notes in Computer Science 7724. Berlin: Springer, 2013.

[33] R Lewison et al. 'Understanding Impacts of Fisheries Bycatch on Marine Megafauna'. en. In: *Trends in Ecology & Evolution* 19.11 (Nov. 2004), pp. 598–604. DOI: 10.1016/j.tree.2004.09.004.

[34] Daoliang Li, Yinfeng Hao and Yanqing Duan. 'Nonintrusive Methods for Biomass Estimation in Aquaculture with Emphasis on Fish: A Review'. en. In: *Reviews in Aquaculture* (Sept. 2019), raq.12388. DOI: 10.1111/raq.12388.

[35] Xiu Li et al. 'Fast Accurate Fish Detection and Recognition of Underwater Images with Fast R-CNN'. In: *OCEANS 2015 - MTS/IEEE Washington*. Oct. 2015, pp. 1–5. DOI: 10.23919/OCEANS.2015.7404464.

[36] J. A. Lines et al. 'An Automatic Image-Based System for Estimating the Mass of Free-Swimming Fish'. In: *Computers and Electronics in Agriculture* 31.2 (Apr. 2001), pp. 151–168. DOI: 10.1016/S0168-1699(00)00181-2.

[37] Rafael López-Leyva et al. 'Comparing Threshold-Selection Methods for Image Segmentation: Application to Defect Detection in Automated Visual Inspection Systems'. en. In: *Pattern Recognition*. Ed. by José Francisco Martínez-Trinidad et al. Vol. 9703. Cham: Springer International Publishing, 2016, pp. 33–43. DOI: 10.1007/978-3-319-39393-3_4.

[38] Patrizio Mariani et al. 'Range-Gated Imaging System for Underwater Monitoring in Ocean Environment'. en. In: *Sustainability* 11.1 (Jan. 2019), p. 162. DOI: 10.3390/su11010162.

[39] Simone Marini et al. 'Tracking Fish Abundance by Underwater Image Recognition'. en. In: *Scientific Reports* 8.1 (Sept. 2018), pp. 1–12. DOI: 10.1038/s41598-018-32089-8.

[40] Fred Mason. 'The Newfoundland Cod Stock Collapse: A Review and Analysis of Social Factors'. en. In: *Electronic Green Journal* 1.17 (Dec. 2002). DOI: 10.5070/G311710480.

[41] Anton Milan et al. 'MOT16: A Benchmark for Multi-Object Tracking'. In: *arXiv:1603.00831 [cs]* (May 2016). arXiv: 1603.00831 [cs]. URL: http://arxiv.org/abs/1603.00831 (visited on 07/03/2020).

[42] H. B. Mitchell. *Image Fusion*. en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-11216-4.

[43] Brendan O'Connor. *F-Scores, Dice, and Jaccard Set Similarity*. Apr. 2012. URL: https://brenocon.com/blog/2012/04/f-scores-dice-and-jaccard-set-similarity/ (visited on 03/06/2020).

[44] *OpenCV: Miscellaneous Image Transformations (getGaussianKernel())*. URL: https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html (visited on 11/06/2020).

[45] Nobuyuki Otsu. 'A Threshold Selection Method from Gray-Level Histograms'. en. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (Jan. 1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076.

[46] Hüseyin Özbilgin. 'Smart Fisheries Technologies for an Efficient, Compliant and Environmentally Friendly Fishing Sector – SMARTFISH H2020'. en. In: *Mediterranean Fisheries and Aquaculture Research* 1.2 (May 2018), pp. 98–99. URL: https://dergipark.org.tr/en/pub/medfar/issue/37150/428551 (visited on 11/06/2020).

[47] Sylvain Paris et al. 'A Gentle Introduction to Bilateral Filtering and Its Applications'. en. In: *ACM SIGGRAPH 2007 Courses on - SIGGRAPH '07*. San Diego, California: ACM Press, 2007, p. 1. DOI: 10.1145/1281500.1281602.

[48] Hitesh A. Patel and Darshak G. Thakore. 'Moving Object Tracking Using Kalman Filter'. In: 2013.

[49] Robert Pilgrim. 'Tutorial on Implementation of Munkres' Assignment Algorithm'. In: Unpublished, 1995. DOI: 10.13140/RG.2.1.3572.3287.

[50] Zhi-Ming Qian et al. 'An Effective and Robust Method for Tracking Multiple Fish in Video Image Based on Fish Head Detection'. en. In: *BMC Bioinformatics* 17.1 (Dec. 2016), p. 251. DOI: 10.1186/s12859-016-1138-y.

[51] Lyle Ramshaw and Robert E. Tarjan. *On Minimum-Cost Assignments in Unbalanced Bipartite Graphs Lyle Ramshaw HP Labs*. Tech. rep. HPL-2012-40R1. HP Laboratories, Oct. 2012.

[52] Dhruv Rathi, Sushant Jain and S. Indu. 'Underwater Fish Species Classification Using Convolutional Neural Network and Deep Learning'. In: *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. Bangalore: IEEE, Dec. 2017, pp. 1–6. DOI: 10.1109/ICAPR.2017.8593044.

[53] *Refractive Index for Some Common Liquids, Solids and Gases*. 2008. URL: https://www.engineeringtoolbox.com/refractive-index-d_1264.html (visited on 12/01/2020).

[54] Petter Risholm et al. 'Real-Time Super-Resolved 3D in Turbid Water Using a Fast Range-Gated CMOS Camera'. en. In: *Applied Optics* 57.14 (May 2018), p. 3927. DOI: 10.1364/AO.57.003927.

[55] Günter Rote. 'Computing the Minimum Hausdorff Distance between Two Point Sets on a Line under Translation'. en. In: *Information Processing Letters* 38.3 (May 1991), pp. 123–127. DOI: 10.1016/0020-0190(91)90233-8.

[56] Yossi Rubner, Leonidas J. Guibas and Carlo Tomasi. 'The Earth Mover's Distance as a Metric for Image Retrieval'. In: *International Journal of Computer Vision* 40.2 (2000), pp. 99–121. DOI: 10.1023/A:1026543900054.

[57] Kenshi Saho. 'Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design'. en. In: *Kalman Filters - Theory for Advanced Applications* (Dec. 2017). DOI: 10.5772/intechopen.71731.

[58] Ahmad Salman et al. 'Fish Species Classification in Unconstrained Underwater Environments Based on Deep Learning: Fish Classification Based on Deep Learning'. en. In: *Limnology and Oceanography: Methods* 14.9 (Sept. 2016), pp. 570–585. DOI: 10.1002/lom3.10113.

[59] Bülent Sankur and Mehmet Sezgin. 'Survey over Image Thresholding Techniques and Quantitative Performance Evaluation'. en. In: *Journal of Electronic Imaging* 13.1 (Jan. 2004), p. 146. DOI: 10.1117/1.1631315.

[60] J. Sauvola and M. Pietikäinen. 'Adaptive Document Image Binarization'. en. In: *Pattern Recognition* 33.2 (Feb. 2000), pp. 225–236. DOI: 10.1016/S0031-3203(99)00055-2.

[61] Raimondo Schettini and Silvia Corchs. 'Underwater Image Processing: State of the Art of Restoration and Image Enhancement Methods'. en. In: *EURASIP Journal on Advances in Signal Processing* 2010.1 (Dec. 2010), pp. 1–14. DOI: 10.1155/2010/746052.

[62] J F Silva, J R Ellis and R A Ayers. *Length-Weight Relationships of Marine Fish Collected from around the British Isles*. en. Tech. rep. 2013, p. 112.

[63] Scott Simms. *Newfoundland and Labrador's Northern Cod Fishery: Charting a New Sustainable Future*. en. Tech. rep. Mar. 2017, p. 42.

[64] *SMARTFISH H2020*. no. URL: http://www.sintef.no/prosjekter/smartfish-h2020/ (visited on 22/02/2020).

[65] Concetto Spampinato et al. 'Automatic Fish Classification for Underwater Species Behavior Understanding'. In: *Proceedings of the First ACM International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams*. ARTEMIS '10. New York, NY, USA: ACM, 2010, pp. 45–50. DOI: 10.1145/1877868.1877881.

[66] Sharath Srini. *The Kalman Filter: An Algorithm for Making Sense of Fused Sensor Insight*. Apr. 2018. URL: https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e (visited on 28/02/2020).

[67] Robin Tillett, Nigel McFarlane and Jeff Lines. 'Estimating Dimensions of Free-Swimming Fish Using 3D Point Distribution Models'. en. In: *Computer Vision and Image Understanding* 79.1 (July 2000), pp. 123–141. DOI: 10.1006/cviu.2000.0847.

[68] C. Tomasi and R. Manduchi. 'Bilateral Filtering for Gray and Color Images'. en. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Bombay, India: Narosa Publishing House, 1998, pp. 839–846. DOI: 10.1109/ICCV.1998.710815.

[69] *Understanding Focal Length and Field of View | Edmund Optics*. URL: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/ (visited on 12/12/2019).

[70] United Nations. *The Sustainable Development Goals Report 2019*. English. New York: United Nations, 2019. URL: https://unstats.un.org/sdgs/report/2019 (visited on 08/05/2020).

[71] Shuo Hong Wang, Jing Wen Zhao and Yan Qiu Chen. 'Robust Tracking of Fish Schools Using CNN for Head Identification'. en. In: *Multimedia Tools and Applications* 76.22 (Nov. 2017), pp. 23679–23697. DOI: 10.1007/s11042-016-4045-3.

[72] Wei Yang, Lulu Cai and Fei Wu. 'Image Segmentation Based on Gray Level and Local Relative Entropy Two Dimensional Histogram'. en. In: *PLOS ONE* 15.3 (Mar. 2020). Ed. by Lixiang Li, e0229651. DOI: 10.1371/journal.pone.0229651.

[73] Fei Yin, Dimitrios Makris and Sergio Velastin. 'Performance Evaluation of Object Tracking Algorithms'. en. In: Jan. 2007, p. 9.

[74] Li Zhang, Yuan Li and Ramakant Nevatia. 'Global Data Association for Multi-Object Tracking Using Network Flows'. en. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, AK, USA: IEEE, June 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587584.

[75] Boaz Zion et al. 'Real-Time Underwater Sorting of Edible Fish Species'. In: *Computers and Electronics in Agriculture* 56.1 (Mar. 2007), pp. 34–45. DOI: 10.1016/j.compag.2006.12.007.