Zebrafish Occlusion Detection

VGIS10 Group 1043

Master Thesis

15/7-2019 to 11/10-2019

Aalborg University
Vision, Graphics and Interactive Systems

This report is compiled in LaTeX. Additionally is Python, Adobe Illustrator, and Inkscape used to code, draw figures, and charts.

Vision, Graphics and Interactive Systems
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Zebrafish Occlusion Detection

Theme:
Computer Vision

Project Period:
15/7-2019 to 11/10-2019

Project Group:
VGIS10 Group 1043

Participants:
Niclas Hjorth Stjernholm

Supervisor:
Malte Pedersen
Thomas B. Moeslund

Number of Pages: 45

Date of Completion:
October 11, 2019

Abstract:

In biomedical research zebrafish (Danio Rerio) serves as a model organism for humans e.g. to test behavioural reactions to various substances. To obtain data for this research the behaviour is observed by recording the motion trajectories of the zebrafish. With multiple zebrafish in an aquarium the tracking of motion of each individual can be complicated by the occurrence of occlusions. The aim of this thesis is to detect the zebrafish occlusions in order to potentially optimise zebrafish tracking.

To detect zebrafish occlusions two solutions are proposed. The two solutions applicability depend on the level of user interaction necessary in a tracking system in order to correct the error caused by occlusions. The first solution is an image classifier, which is utilised to classify if an image contains an occlusion. The second solution is a Faster Region-based Convolutional Neural Network (R-CNN) based object detection, which is utilised to localise and categorise occlusions, based on six pre-defined categories, in the image.

A novel categorisation of zebrafish occlusions is presented together with an implementation of a proof of concept multi-class object detection for detecting multiple zebrafish occlusions.

Preface

This report is composed by Niclas Hjorth Stjernholm as the final project of the Master's Programme in Vision, Graphics and Interactive Systems at Aalborg University. The focus of the thesis is to design and implement af computer vision system able to detect when zebrafish occlude each other in an image.

Reading Guide

Figures and tables are numbered sequentially within each chapter. All references in the thesis is a hyperlink and can be followed to where it is referencing. For citation the report employs the Harvard method. If citation is not present in tables or figures, they are produced by the author.
This project is implemented in Python 3.6 using the Jupytor Notebook available on Google Colaboraty.

Aalborg University, October 10, 2019

Niclas Hjorth Stjernholm
<nstjer14@student.aau.dk>

Contents

Glossary

CNN	Convolutional Neural Network.
COCO	Common Objects in Context.
FC	Fully Connected.
FPS	Frames Per Second.
IoU	Intersection over Union.
mAP	mean Average Precision.
NIR	Near Infra-Red.
NMS	Non-Maximum Suppression.
R-CNN	Region-based Convolutional Neural Network.
ReLU	Rectified Linear Unit.
ROI	Region of Interest.
RPN	Region Proposal Network.
SVM	support vector machines.

Chapter 1

Introduction

In the world of biological and medical research, tests and experiments are performed on animals before any applications to humans can be done. This is due to both ethical aspects, feasibility, and potential harm and fatalities. The type of animals used in research are known as model organisms, as they often model the biology of humans. An example of a model organism is mice, which are often used to study medical treatments for humans, as they share some genetic sequences with humans [Perlman, 2016; Khan and Alhewairini, 2018]. In general, a model organism is not just animals relatable to human genetics, but multiple different kinds of organisms, such as fungi and plants. The requirement is them having a similarity to the organism they are modelling [Hedges, 2002].

Another example of an animal model is the zebrafish (Danio Rerio). Zebrafish and humans share a lot of pathways which control development of the central nervous system, and since the embryo of the zebrafish is clear, it enables observation of the development in the early stages of life. Zebrafish also has a 70% disease gene similarity to humans.

The zebrafish is often used as an animal model, not only due to the similarities to humans, but also due to size and cost. The zebrafish is much smaller than a mouse and the maintenance of the animal is lower due to this, and more zebrafish can be kept on the same amount of space than mice. The adult zebrafish can lay up to 200 eggs each week if kept in optimal conditions, which means acquirement of new test subjects will not present it self as an issue.

While being a good model organism for humans, the zebrafish is also used as a model organism for aquaculture species in areas such as, development, diseases, and behavioural tendencies among aquaculture [Khan and Alhewairini, 2018].

When testing or experimenting on zebrafish they are often video recorded to be able to observe them. To automate the process of recording the zebrafish, computer vision is often employed to obtain as much data from the recordings as possible, as doing this manually is a demanding task.

Chapter 2

Analysis

2.1 Behavioural Analysis of Zebrafish

Zebrafish are very social animals and have tendencies to form groups [Khan and Alhewairini, 2018]. An aggregation of zebrafish can be either a shoal or a school depending on whether the zebrafish are interacting socially or not. When an aggregation of zebrafish is due to social interaction, the zebrafish are shoaling and the swimming pattern is chaotic, but when zebrafish are schooling, the swimming pattern is tightly coordinated and organised [Miller and Gerlai, 2012]. Identification of the swimming pattern can thereby be studied to investigate social phenotypes and behavioural patterns when affected by a certain drug or pheromone [Khan and Alhewairini, 2018].

Kalueff et al. [2013] describes multiple scenarios in which the zebrafish is prone to sudden changes in both speed and direction. Depending on the situation, the zebrafish can move very erratic often reflecting a state of anxiety or fear. Furthermore, sudden bursts can also occur in an attempt to attack another zebrafish, often connected to social dominance [Kalueff et al., 2013].

As mentioned, the zebrafish is also used to test the general behaviour when affected by a drug. This is done by having a control group of zebrafish and recording the normal behaviour in an aquarium and then comparing this behaviour with a drug affected behaviour [Stewart et al., 2015]. Figure 2.1 shows an example of swimming patterns of zebrafish, under influence of different drugs, made by Stewart et al. [2015]. It is clear how the drugs changes the swimming patterns of a zebrafish, which may be transferable to how it will affect humans as well.

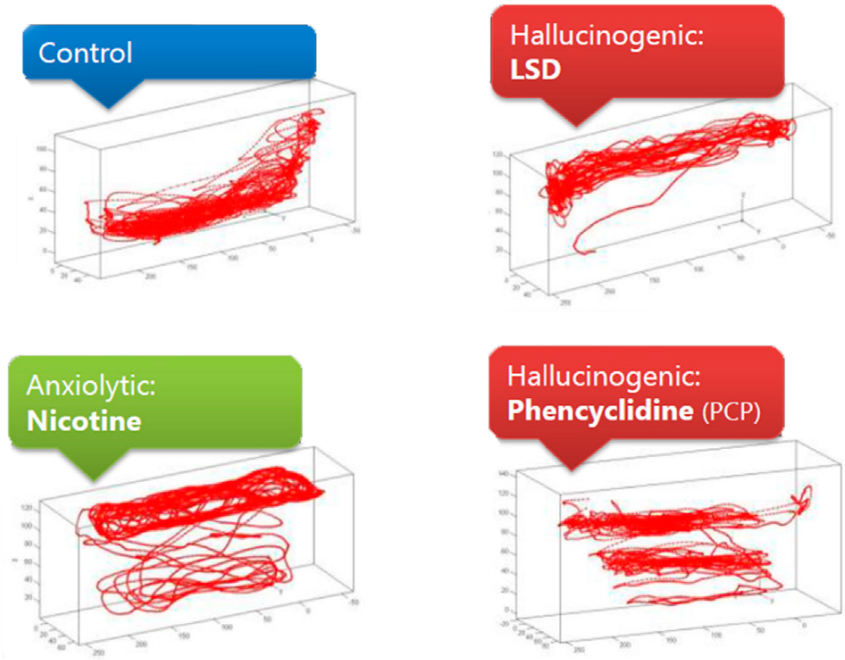


Figure 2.1: Zebrafish swimming behaviour under influence of drugs, by Stewart et al. [2015]

2.2 Tracking of Zebrafish

To investigate the difference in behavioural patterns, i.e. due to drugs, each zebrafish must be detected individually over time. Detection of the zebrafish is often done with an automated tracking system, as done by Stewart et al. [2015], as manual annotation of the data often proves infeasible.

2.2.1 Tracking Requirements

In order to be able to track zebrafish, collection of data is firstly necessary. Recording of the zebrafish is done using one or more cameras either from the top of the aquarium or looking into the aquarium from the side.

Due to the erratic movement of the zebrafish, the higher the Frames Per Second (FPS) the camera is able to capture video at, the better, as this will increase the probability of the camera capturing every movement of the zebrafish. Bengtson and Pedersen [2017] states, that even at 240 FPS the zebrafish is still somewhat blurred when accelerating, which can lead to a lower detection rate.

The top down view of the aquarium when recording, is often used when using a single camera setup. The top view takes advantage of the uniform and rigid head physiology of the zebrafish. Even though the zebrafish is able to contort its body, the head most often remains the same, when using the top view. Examples of the rigid head is shown in Figure 2.2.

Figure 2.2: Examples of the rigid head of the zebrafish in different grades og contortions of the body

The data shown in Figure 2.2, is captured from above the aquarium with a Near Infra-Red (NIR) backlight beneath the aquarium.

When recording from the side of the aquarium more details of the zebrafish is available, as the zebrafish has uniquely coloured stripes on both sides, which can be used to identify the zebrafish [Karpova and Haurum, 2018].

According to Qian and Chen [2017], due to the shape of the zebrafish, it generally takes up more space when filmed from the side than from a top view. Furthermore, when the zebrafish turns towards or away from the camera, the shape will be very different than when looking at the side of the zebrafish [Bengtson and Pedersen, 2017]. Examples of both a regular side view and some issues from the view is shown in Figure 2.3.

Figure 2.3: Examples of positions of zebrafish in the side view. Image from Bengtson and Pedersen [2017]

When data is acquired, it will need to be prepared before tracking or annotating is done.

2.2.2 Annotating Data

Annotating data is understood as manually marking each zebrafish/object in every frame of a video, whereas, when using an automated solution it can also be referred to as tracking system.

A tracking system can be split into multiple steps:

- Pre processing
- Detection
- Create trajectories

Pre Processing

When making operations on a video, the file is split into individual frames, and every frame is treated as an individual image. Before locating of the zebrafish in the image is done, some pre processing of the image is often performed. This often includes removing the background and noise from the image.

This is done to ease the process of locating the zebrafish as it will be isolated in the image.

Detection

Detection of the zebrafish is done in multiple ways. A detection will ultimately produce a single point in the image.

Head detection of the zebrafish is an often used approach due to the rigid head of the zebrafish. This means the head will keep the same shape while swimming, whereas the rest of the body may change shape, which will make it harder to detect if focus is on the entire zebrafish.

Other examples of detection, are centre of mass of the zebrafish or extracting a skeleton of the zebrafish, representing the shape of the object with a line. Finding the centre of mass of the zebrafish, if not being limited, may end up outside of the object if it is bending into a shape looking like a C.

Examples of extracted points from detections are shown in Figure 2.4.

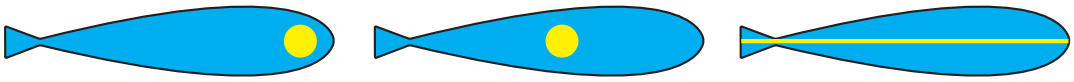


Figure 2.4: Examples of extracted points from detections. The first zebrafish is head detection, the second is centre of mass, and the third is skeletonisation

Create Trajectories

When the desired point in the zebrafish is extracted, the point needs to be linked together with points in other images to create a trajectory. If only a single zebrafish is present in the aquarium no identification is necessary, as every point found belongs to one individual.

As soon as multiple zebrafish are present in the aquarium at the same time, a decision needs to be made to connect the previous frame's detections to the new ones in the current frame. This can be done by predicting where each individual will be in the following frame based on a state vector and experience from previous frames as input to a Kalman filter, which then makes a qualified guess based on statistics of the new positions. Besides a Kalman filter, a simple cost function such as the Hungarian algorithm can be applied to link detections.

An issue occurring when multiple zebrafish are in the same aquarium, is when two or more individuals lie close enough together or on top of each other, which may confuse or trick the prediction and cost algorithm.

2.2.3 Occlusions

An occlusion is when one object is hidden or overlapped by another object from a specific point of view e.g. from the camera view. When a zebrafish occlude another there is a risk of losing the detection and thereby the position of the occluded zebrafish in one or more concurrent frames. According to Green et al. [2012] the use of automated tracking systems perform with same accuracy as manual annotations but in a faster manner. However, they state that automated tracking systems have complications with occlusions.

When a detection is lost due to an occlusion, the identity of the zebrafish may be lost as well. If no re-identification is employed in a tracking system, a new ID may be assigned to the object which was lost due to an occlusion. This scenario is visualised in Figure 2.5

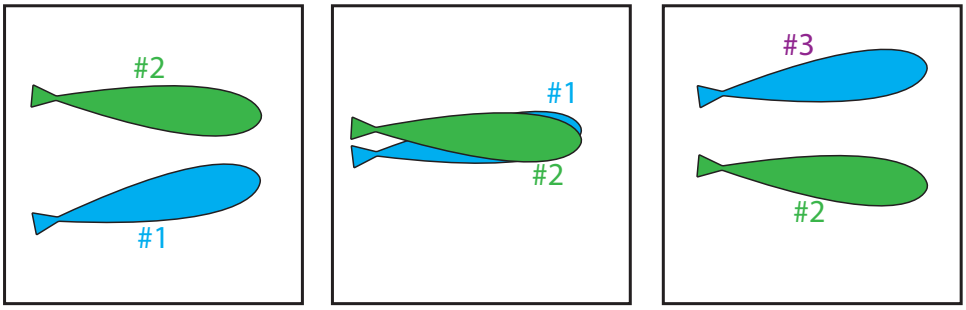


Figure 2.5: Re-ID scenario due to occlusion

Not all occlusions will cause the same types of complications, and some occlusions do not cause any disruption of the trajectory. This is determined by the detection system deployed to track the zebrafish. If the detection of a zebrafish is centred at the head, no occlusion will be detected when only the bodies of two zebrafish overlap, however, if detection is done by either skeletonisation or centre of mass, an occlusion will most likely occur [Feijó et al., 2018]. An example of this scenario is shown in Figure 2.6.

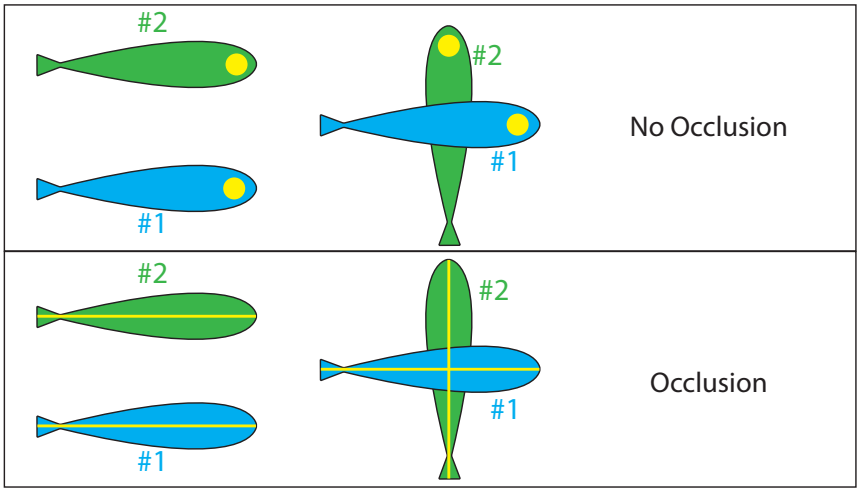


Figure 2.6: Different types of detection leads to different types of occlusion

Missing detections and wrong identifications are undesirable, as this will require a user of the tracking system to intervene and correct the errors, which will prolong the process. Therefore, solutions to either handling the occlusions or automatically solving the wrong identification are often implemented in a tracking solution.

2.2.4 Object Detection

In order to be able to detect the zebrafish occlusions in every frame, object detection is necessary. Object detection is defined as an employment of computer vision which recognises unique features of objects in an image.

Object detection consists of two main tasks; classification and localisation. It can furthermore be split into two categories depending on the amount of classes which must be detected. If it is only one class, such as detecting humans in an image or detecting any kind of zebrafish occlusion, it is known as class-specific detection. Whereas detecting multiple different kinds of objects is known as multi-class detection [Zhang et al., 2013].

The main objective of an object detector is to generate a label list of predefined objects detected in an image. The list should specify which classes are present and where in the image they are located.

Deep Learning

When using deep learning and a Convolutional Neural Network (CNN) for object detection one of the earliest implementations was the R-CNN. As the name suggests, the method is based upon a region proposal based CNN instead of using the often previously used sliding window technique which some times can lead to a large amount of testing windows.

The CNN used in the R-CNN is utilised for feature extraction feeding into a class-specific support vector machines (SVM) to categorise each object. The CNN is pre-trained on the ImageNet database and then trained on the PASCAL VOC dataset. An overview of the R-CNN pipeline is shown in Figure 2.7. More in-depth of deep learning is found in chapter 5.

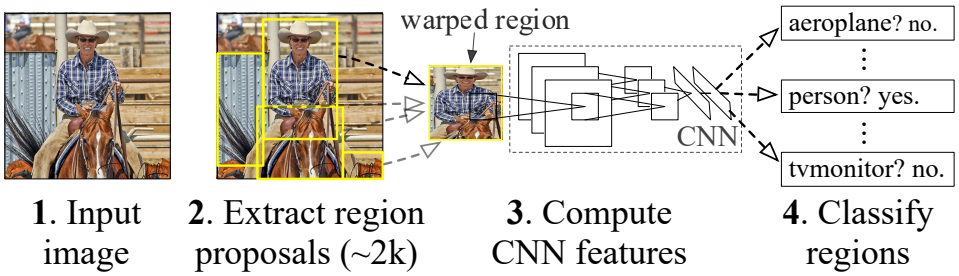


Figure 2.7: Simple flow diagram of the R-CNN by Girshick et al. [2014]

As shown in Figure 2.7 the R-CNN is split into 3 individual steps consisting of; region proposals, feature extraction, and SVM classification.

The region proposals are extracted using *SelectiveSearch* which initialises regions in the image and then merge these with a hierarchical grouping [Uijlings et al., 2013]. The final grouping is then a box containing the entire image. The regions are grouped in relation to colour space and similarity [Girshick et al., 2014].

The next step in the solution is warping the proposed regions to fit the CNN input size and extracting features from these and produce a 4096-dimensional feature vector. The feature vectors consist of both positive and negative proposals found with the SelectiveSearch. These vectors are used to train the class specific SVMs, one SVM per class in which a background class is included [Girshick et al., 2014]. When testing a given image, the SelectiveSearch generates approximately 2000 proposals, which each are propagated through the CNN to extract feature vectors. Each feature vector is tested against every class-specific SVM. To remove overlapping detections a greedy Non-maximum Suppression is applied [Girshick et al., 2014].

At the time of publication, the R-CNN achieves state of the art performance in object detection, with a 13% increase in precision compared to the previously best method [Girshick et al., 2014].

The R-CNN does leave room for improvements as it is a slow solution when testing on an image. Furthermore, as the solution is split into multiple modules, the loss calculations for training the SVMs is not used to update the CNN.

An improvement on both speed and accuracy of the R-CNN was published a year later, called Fast R-CNN. Instead of using the multi module pipeline, as illustrated in Figure 2.7, training is now done end-to-end. The solution takes an image and a set of pre-computed object proposals, like the R-CNN [Girshick, 2015].

Instead of individual proposals as in the original R-CNN, the new iteration propagates the entire image forward through multiple convolutional and max-pooling layers to produce a feature map. The features are extracted from each proposal using a Region of Interest (ROI) pooling layer. After the ROI pooling layer follows two fully connected layers leading to two different outputs; a softmax classification layer and a bounding box regression layer [Girshick, 2015]. This pipeline is also shown in Figure 2.8.

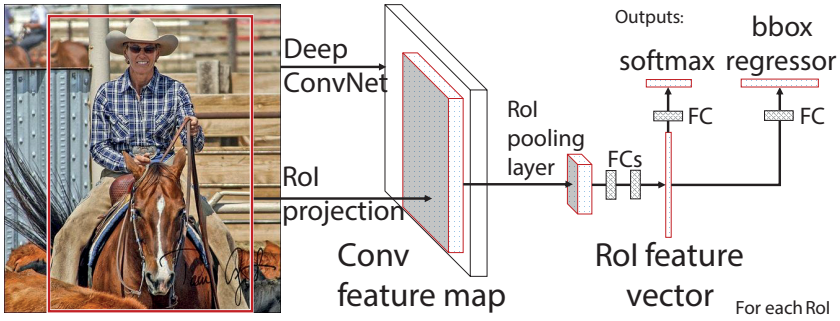


Figure 2.8: Fast R-CNN pipeline overview, by Girshick [2015]

Just like the R-CNN, the Fast R-CNN is pre-trained using the ImageNet database fine tuned for object detection, but instead of being based on AlexNet, as R-CNN is, the best performing solution of Fast R-CNN is based on the deeper VGG16 network [Girshick, 2015]. When comparing to the previous iteration, R-CNN, it is done while both solutions are using the VGG16 network layout instead of the AlexNet for the R-CNN. This is done to have a common ground for comparison. Fast R-CNN improves precision with 4% [Girshick, 2015].

The main improvement of the new model is more in speed both in training and testing due to the computing of a convolutional feature map for the entire image. The speed increase is only in relation to the actual detection, as the region proposals are slow and create a bottleneck for the system [Girshick, 2015].

To combat the slow region proposals of Fast R-CNN, Ren et al. [2017] created a third iteration of the R-CNN network called *Faster R-CNN*. Here a network named Region Proposal Network (RPN) is implemented to compute region proposals as part of a network. The RPN shares the convolutional layers and the feature map with the ROI pooling in the Fast R-CNN. Due to the layers being computed on the entire image, the time used for proposals generated using the RPN is much lower than of a method such as SelectiveSearch. The RPN, computing region proposals, is the only new part of the Faster R-CNN compared to Fast R-CNN.

The results of the Faster R-CNN in precision are rather small, but the entire computing time of the solution has been significantly minimised, from about 2 seconds per image to 0.2 seconds, including region proposals for both timings.

Multiple solutions on the Common Objects in Context (COCO) dataset object detection leader board are still employing the framework of R-CNN and its predecessors, which shows why the R-CNN is still a state of the art solution.

2.2.5 Handling Occlusions

According to a study by Qian and Chen [2017], occlusions will occur both while recording from above and from the side of an aquarium, but with greater occurrence from the side of the aquarium due to the shape of the zebrafish. As previously mentioned, occlusions can cause errors, resulting in missing data for an individual due to a new ID [Feijó et al., 2018].

A solution to missing tracking data due to occlusions, is to re-link parts of the trajectories (tracklets) to create complete trajectories. This can be done by computing a state vector for each zebrafish and using a Kalman filter which makes predictions of the zebrafish's position, and thereby estimate what ID belongs to the different zebrafish after an occlusion [Feijó et al., 2018; Qian et al., 2014].

To avoid patching in missing data, another more feasible solution could be made to solve occlusions before they occur by detecting the zebrafish in each frame. Both Romero-Ferrero et al. [2019] and Dolado et al. [2014] propose solutions which detect occlusions in an effort to solve them using computer vision. Dolado et al. [2014] has categorised occlusion types by how the zebrafish overlap each other in an effort to specify the solution. However, the only way they solve the occlusions are through a two-step trial and error process i.e. if the first step does not solve the solution, the second step is applied, without factoring in the occlusion type. However, a novel approach could be to recognise an occlusion type in order to apply a predefined optimal solution.

2.3 System Dependencies

Different types of tracking systems may vary in requirement of user interaction throughout the tracking. The ability of the solution may require the user to interact with the system when occlusions occur in the video data in order to pick the correct path for each zebrafish and thereby solve or fix the data.

If a solution has a high level of user interaction, localisation of an occlusion may not be a requirement due to the necessary interactions of a user, but recognition of an occlusion occurring in the image may be sufficient.

With this analysis a specification of the problem, sought out to solve, is presented in the following chapter.

Chapter 3

Problem Statement

Based on chapter 1 and chapter 2, a problem statement is made:

How can a zebrafish occlusion detection solution be implemented in order to potentially optimise zebrafish tracking?

As stated in section 2.3 the kind of detection necessary is dependent on the system. Therefore some specifications of the problem statement are defined with the following questions:

- How can zebrafish occlusions be categorised if necessary?
- How can occlusions be detected in a sequence of images of zebrafish using computer vision?
- What kind of detection is required based on user interaction level?

Chapter 4

Data Analysis

To be able to develop a solution which is able to detect which type of occlusion occurs and the location of it, some analysis of the data is necessary. This is also done to determine how many and which kind of categories of occlusions there are.

4.1 Video Capture

The video used is recorded from above the aquarium with a NIR backlight which makes the dark zebrafish stand out in contrast to the background. The video is 60 seconds long with three zebrafish recorded at 50 FPS yielding a total of 3000 frames to be processed. An overview of the recording setup is shown in Figure 4.1

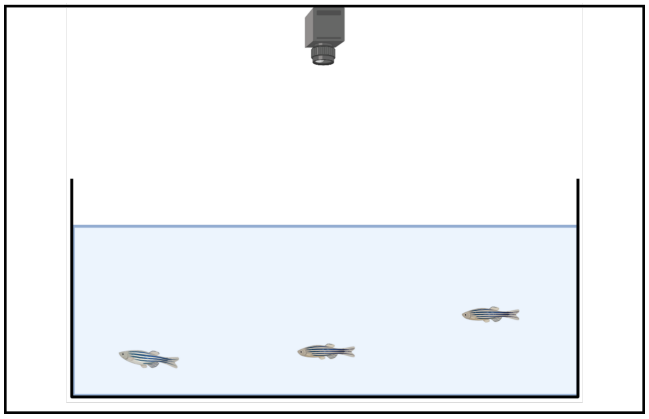


Figure 4.1: Video recording setup

4.1.1 Annotating Data

The data is annotated with bounding boxes for object detection. Annotations are made using the *labelme* Python annotation tool [Ketaro, 2016]. A snapshot of the UI of the software is shown in Figure 4.2.

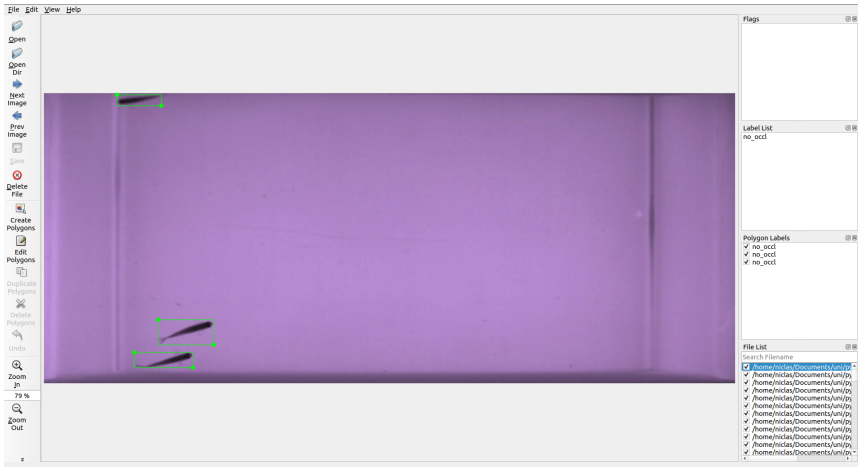


Figure 4.2: Snapshot of labelme annotation tool while annotating data

The software is able to export annotations in formats matching both VOC PASCAL and COCO, which is either an XML-file for each image using the VOC PASCAL annotation style or a json file using the COCO annotation style. The file-type chosen is the PASCAL VOC XML-files, with individual files for each image annotated.

The annotations are made up of both *no occlusions* and different types of occlusions presented in the following section.

4.2 Fish Shapes

Even a single zebrafish is able to twist into many different shapes, according to Kalueff et al. [2013] zebrafish has an extensive catalogue of behavioural patterns, which are also based on different contortions of the body. The main shapes of a single zebrafish are shown in Figure 4.3. The shapes shown are only the ones seen from the top view of an aquarium.

(a) Zebrafish lying completely straight

(b) Zebrafish creating a C shape

(c) Zebrafish creating an S shape

Figure 4.3: Different shapes a zebrafish makes from a top view, also described by Kalueff et al. [2013]

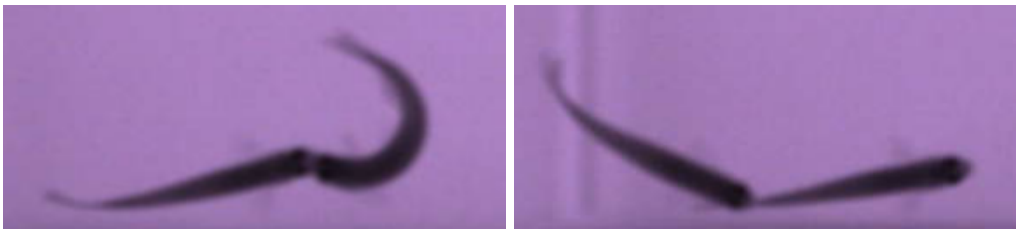
Kalueff et al. [2013] states that the C-shape is made as starting movement to propel itself forward, often in order to escape or because the zebrafish is startled. The

bending into a C-shape, can also be caused by the zebrafish turning. The S-shape is made for the same reasons as the C-shape, to propel the zebrafish forward.

When two zebrafish are touching or crossing each other in any way, they create multiple new shapes.

4.2.1 Touching

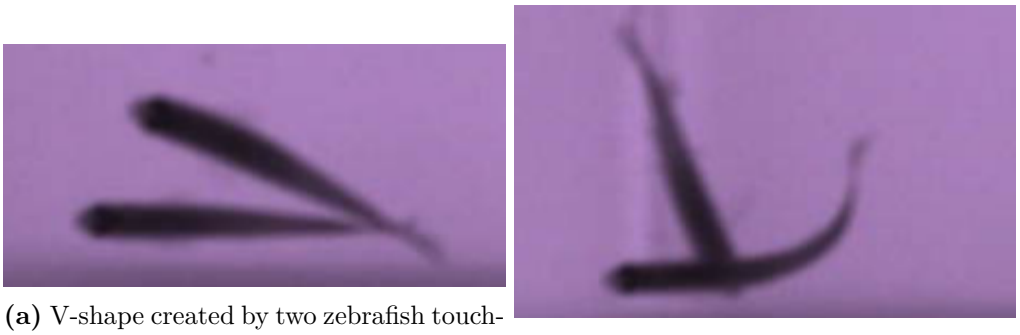
When two zebrafish are touching they in general create three different kind of shapes. They will either be elongating one another, creating one single long object. This can be either head-to-head, tail-to-tail, or head-to-tail. Two examples are shown in Figure 4.4. When touching head-to-tail it is most often seen as courtship between a male and a female zebrafish [Kalueff et al., 2013].



(a) Head-to-head elongation **(b)** Head-to-tail elongation

Figure 4.4: The two touching possibilities creating an elongation

Two zebrafish touching is done in all kinds of positions and angles. When touching end-to-end but at a tighter angle, two zebrafish tend to create a V-shape. In the beginning and end of a crossing or when touching the head or tail to the middle of the body of another zebrafish, a T-shape is created. Examples of both shapes are shown in Figure 4.5.



(a) V-shape created by two zebrafish touching tails

(b) T-shape example

Figure 4.5: Examples of V- and T-shape

4.2.2 Above and Below

More complex occlusions often happen when one zebrafish is above the other. The water depth of the aquarium affect this as the zebrafish are more likely to swim above each other if the water depth allows for it. There is mainly two different occlusions happening in this instance; two zebrafish creating a cross or one zebrafish almost completely occluding the other.

The crossing can happen in multiple angles. A crossing occlusion is defined as two zebrafish creating an X-shape. Examples of different crosses are shown in Figure 4.6.

Figure 4.6a shows a crossing occlusion, which is close to being an *on-top* occlusion, but will be defined as a cross due to the head and tails being separated.

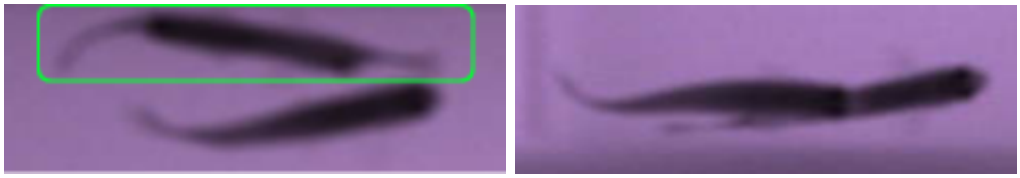
(a) Cross occlusion at a wide angle

(b) Cross occlusion at a semi-wide angle

(c) Cross occlusion at an almost right angle

Figure 4.6: Different cross occlusions

The on-top occlusions is when one zebrafish is somewhat straight on top of another zebrafish. This kind of occlusions can, at times, be a shorter kind of elongation e.g. when the head of the top zebrafish is further in on the body of the bottom zebrafish. Examples of op-top occlusion are shown in Figure 4.7.



(a) Straight on top occlusion of two zebrafish

(b) Head on top of middle of bottom zebrafish

Figure 4.7: On-top occlusions by two zebrafish

4.2.3 Other Occlusions

Besides the five previously mentioned types of occlusions, several other occlusions occur which are harder to classify as they do not fall into any specific category. Due to this, a category named *other* is created as well. Some examples of these kinds of occlusions are shown in Figure 4.8.

(a) Example of occlusion categorised as *other*

(b) Example of occlusion categorised as *other*

(c) Example of occlusion categorised as *other*

Figure 4.8: Different *other* occlusions

Occlusions occurring with all three zebrafish are also categorised as other, as none of these fall into the defined categories, but are still occlusions.

The more zebrafish included in an occlusion the higher the complexity of the occlusion is prone to be. This includes both the type of occlusion occurring between each pair of the zebrafish but also possible occlusions and crossings of multiple zebrafish occurring at the same time. At some occasions, a three zebrafish occlusion is easily split into two different types of occlusions and all three zebrafish are very distinguishable from one another, as in Figure 4.9a, but at other times three zebrafish will stack upon each other as in Figure 4.9b.

(a) Three zebrafish in distinguishable occlusions

(b) Three zebrafish in a stacking occlusion

Figure 4.9: Occlusions amde by three zebrafish

As can be seen in Figure 4.9, the occlusions occurring are either combinations of the occlusions with only two zebrafish or the same type of occlusion multiple times.

Due to this it is chosen to focus on the occlusions occurring between two zebrafish and prove feasibility of this to be able to, later, expand a solution to handle occlusions with more than two zebrafish.

4.3 Occlusion Frequency

In the data set the occlusions happen with different frequencies. The frequencies are displayed in a bar chart in Figure 4.10.

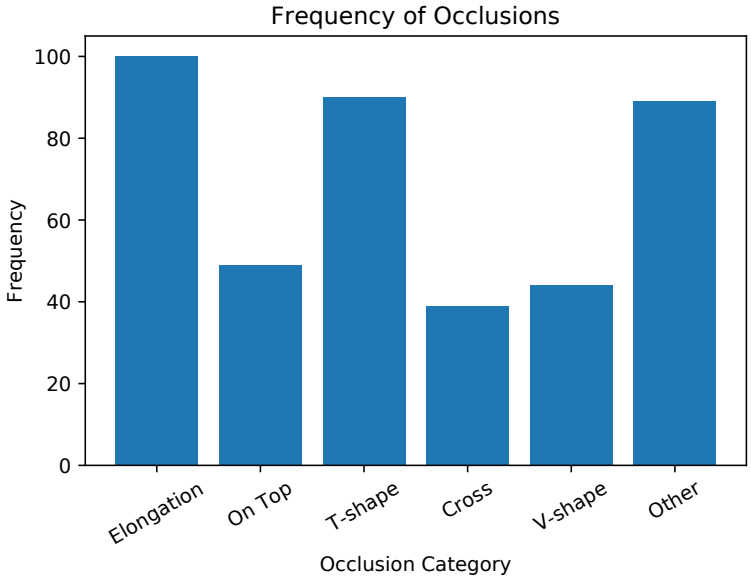


Figure 4.10: Frequency of the different types of occlusions

Figure 4.10 shows that elongation is the most frequently occurring occlusion, together with the T-shape and other types of occlusions. In total 411 occlusions occur out of 3000 frames, which is a frequency of 13.7%.

Chapter 5

Design

Based on the acquired knowledge from chapter 2 and chapter 4, a design of occlusion detection solutions are made. This is implemented using Python3 with the Keras API working with Tensorflow.

This chapter gives an in-depth description of how the implementations are made and the theory behind them.

As stated in chapter 2 the use of CNNs in object detection is the highest performing solution in different benchmarks. Due to this it is chosen to utilise CNNs to perform the occlusion detection of zebrafish. In the following, a description of CNNs in general and of the implementation for occlusion detection is presented.

5.1 Convolutional Neural Networks (CNN)

As regular neural networks, CNNs are made up of neurons which have learnable weights and biases. Each neuron in the network receives inputs, performs an operation on the input and passes the output on, either linearly or non-linearly, dependent on the design.

A CNN receives an input image and then transforms this through a series of hidden layers in the network. Each hidden layer consist of a set of neurons, and each neuron is fully connected to the neurons in the previous layer, but does not share any connections inside the layer. The last layer, the output layer, is a fully connected layer and represents the class scores. A CNN has neurons arranged in three dimensions, namely: width, height, and depth. Already in the input an RGB image will have a depth of three due to the three colour channels [Karpathy, 2016]. A small example of the CNN architecture is shown in Figure 5.1.

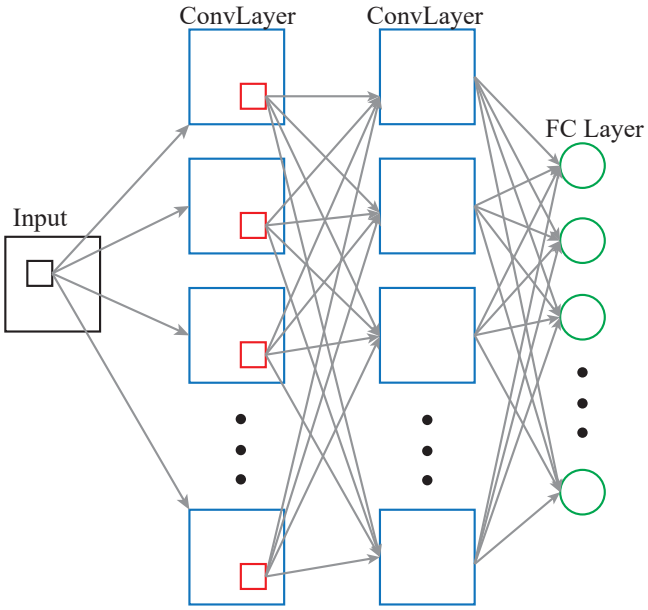


Figure 5.1: Example of a CNN architecture

5.1.1 Layers

A simple CNN will often consist of a sequence of layers, in which each layer transforms one volume of activations to another through differentiable functions. The main layers used are; convolutional layer, pooling layer, and fully connected layer. These layers are presented in the following.

Convolutional Layers

The convolutional layer is the main layer of operations in a CNN, hence the name of the network type.

A convolutional layer operates using a filter also known as a kernel to compute an output by iterating through the input image. This filter convolutes the input using the kernel weights to calculate a dot product of the input. Each filter produces a 2-dimensional activation map. If multiple filters are applied, the 2D activation maps will be stacked along the depth dimension in the output .

Besides being able to control the width and height of the kernel, most often a square, it is also possible to control the depth. Depth is a hyper parameter and corresponds to the amount of filters applied to the input. The neurons along the depth dimension may activate on different edges or colours, but are still focusing on the same region as the rest of the depth column.

When sliding a kernel over an input, a stride can be selected. With a stride of 1 the filter moves only one pixel at a time, while with a stride of 2 or higher the filter will move multiple pixels before performing a convolution again.

To enable a kernel to compute all pixels of an input, zero padding is necessary. The size of the zero padding is another hyper parameter, and it allows for controlling the size of the output, mostly for keeping the input size, in height and width, in the output [Karpathy, 2016]. A one dimensional example of the influences the different hyper parameters has is shown in Figure 5.2.

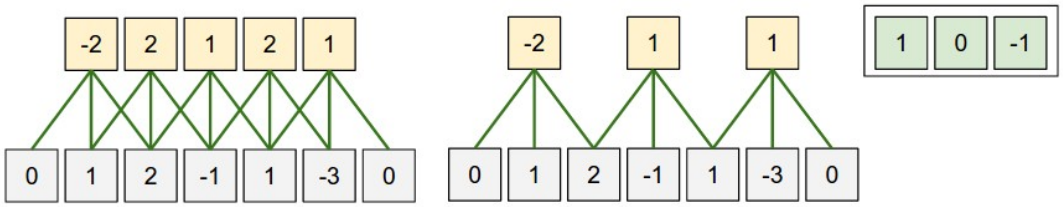


Figure 5.2: With a kernel of [1, 0, -1] the left example shows an input in the bottom with zero padding of size 1 and stride of 1. The right convolution has changed to stride 2 [Karpathy, 2016]

Figure 5.2 shows that with a stride of one and employing zero padding, it is possible to keep the dimensions of the input in the output.

The output of a kernel can be calculated using the Equation 5.1:

$$(W - F + 2P)/S + 1 \tag{5.1}$$

Where W is the input volume size, F is the kernel size, P is the amount of zero padding used on the border of the input and, S is the stride. Using the example in Figure 5.2, the output can be calculated:

$$(5 - 3 + 2 \cdot 1)/1 + 1 = 5 \tag{5.2}$$

The application of stride has some limitations, as the result of Equation 5.1 has to be an integer [Karpathy, 2016].

Pooling Layers

To be able to control overfitting and to reduce the amount of parameters thereby the computations in a network, pooling layers are often periodically placed in-between successive convolutional layers. A pooling layer reduces the spatial size of the input.

The size reduction is done using a filter in the same way as with a convolutional filter, but no convolutions are performed, instead, dependent on the kind of pooling, a value inside the filter is chosen to be passed on to the ooutput map. Using the most common type of pooling, Maxpooling, often with a filter size of 2×2 and a stride of 2, the highest pixel value within the filter is chosen as the output [Karpathy, 2016]. An example of this is shown in Figure 5.3.

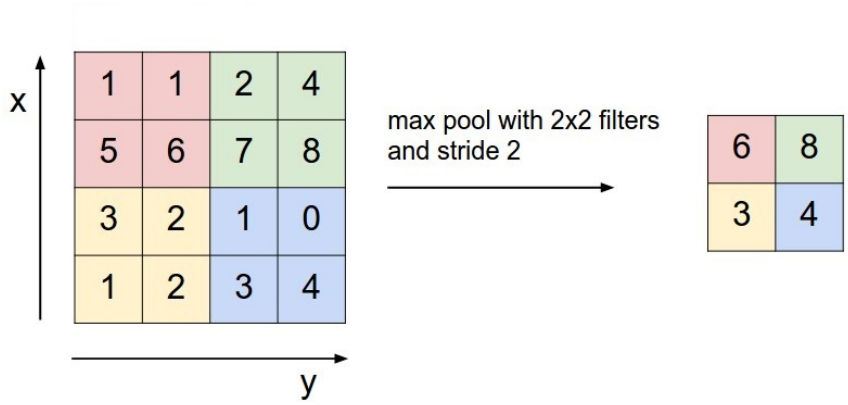


Figure 5.3: Maxpooling example with a 2×2 kernel and stride 2 [Karpathy, 2016]

Classification

To be able to use the features extracted from input image in the convolutional layers one or more Fully Connected (FC) layers are added in the end of a CNN. As the name suggests, all the neurons in the previous layer are connected to the FC layer.

To classify N number of classes, the last FC layer has the same amount of neurons as the amount of classes. The output of the last FC layer is then the probability of the input image belonging to each class, with a sum of all output neurons being 1 [Karpathy, 2016].

5.2 Image Classification

A simple occlusion detection can be made without any localisation in order to notify a system of an occlusion present in an image. This is done using an image classification network, trained on two classes; *occlusion* or *no occlusion*.

The network chosen is the VGG16 network, as this is also what the R-CNN classification network is based upon. An overview of the layers in the VGG16 network is shown in Figure 5.4.

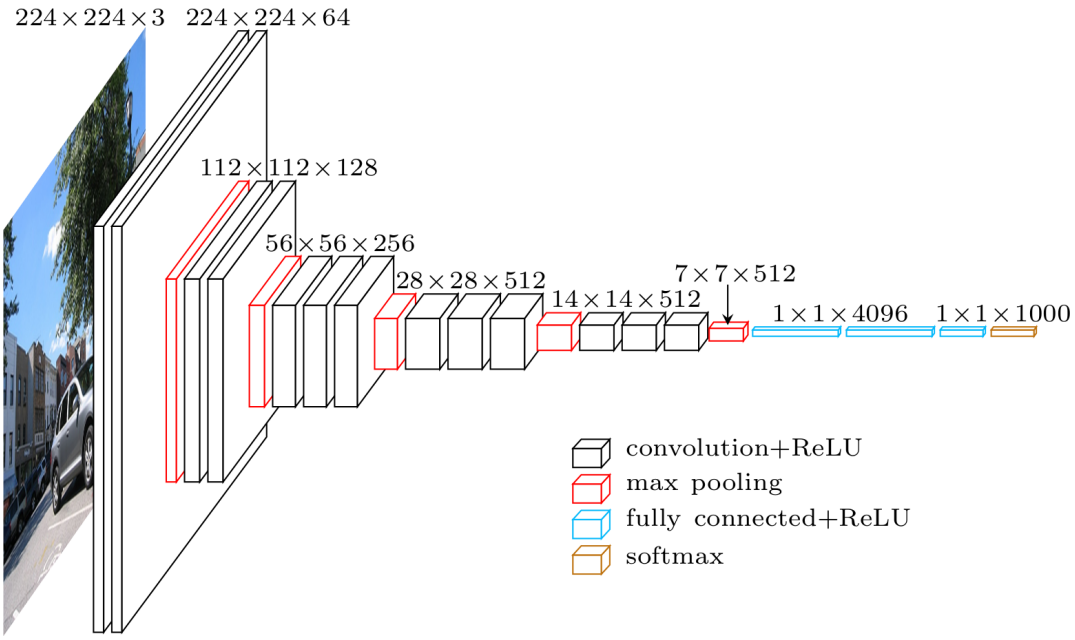


Figure 5.4: VGG16 structure

The network is trained with the pre-trained weights from the ImageNet database. The network can be split into five blocks of convolutions with a max pooling layer after each block.

The images fed into the network are of size 224×224 and the last fully connected layer has a depth of 2 due to the amount of classes. The structure of the network is shown in Table 5.1. Table 5.1 also shows an extra dense layer has been added, and the depth of the FC layers has been made smaller.

Table 5.1: Detailed description of VGG16 design. The input is $224 \times 224 \times 3$ images

Layer Type	Feature Map Size	Kernel/Pool Size	Activation
Block 1			
Conv2D	64	3×3	ReLU
Conv2D	64	3×3	ReLU
MaxPooling2D		2×2	
Block 2			
Conv2D	128	3×3	ReLU
Conv2D	128	3×3	ReLU
MaxPooling2D		2×2	ReLU
Block 3			
Conv2D	256	3×3	ReLU
Conv2D	256	3×3	ReLU
Conv2D	256	3×3	ReLU
MaxPooling2D		2×2	
Block 4			
Conv2D	512	3×3	ReLU
Conv2D	512	3×3	ReLU
Conv2D	512	3×3	ReLU
MaxPooling2D		2×2	
Block 5			
Conv2D	512	3×3	ReLU
Conv2D	512	3×3	ReLU
Conv2D	512	3×3	ReLU
MaxPooling2D		2×2	ReLU
Classification			
Flatten			
Dense	1024		ReLU
Dense	1024		ReLU
Dense	512		ReLU
Dense	Amount of Classes		Softmax

5.3 Faster R-CNN Object Detection

To be able to localise the occlusions in the image and mark them with bounding boxes, an object detection solution is necessary. For this the Faster R-CNN presented in chapter 2 is implemented. In the following the different objects in the pipeline are described in more detail. An example overview of the pipeline is shown in Figure 5.5.

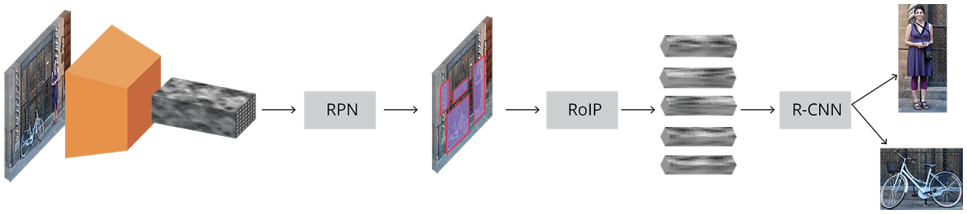


Figure 5.5: Overview of the Faster R-CNN pipeline and its steps [Rey, 2018]

5.3.1 Region Proposal Network

As mentioned in section 2.2.4 Object Detection, the Faster R-CNN utilises a network called *Region Proposal Network (RPN)* to produce region proposals. The RPN shares some convolutional layers with the CNN. By sliding a small network over the feature map produced by the last shared convolutional layer [Ren et al., 2017].

As the objective of the object detection is to find bounding boxes in the image, anchors are used. Anchors are fixed bounding boxes of different sizes and ratios and are placed throughout the image. They are used for reference when predicting object locations. An anchor is created for each point in the produced feature map, but the final anchors still reference the input image of the network. The RPN outputs a set of good proposals based on the anchors, by having two outputs for each. The first output is an object probability score, the second is bounding box regression for adjusting the anchors to fit potential objects. An example of how anchors are placed in an input image is shown in Figure 5.6.

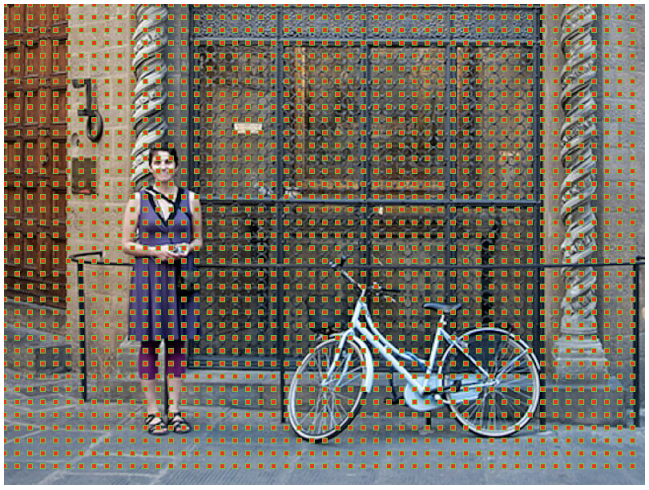


Figure 5.6: Example of anchor centres placed in an input image [Rey, 2018]

For training the RPN, the anchors are split into two different categories; the first are those which overlap a ground-truth object with an Intersection over Union (IoU) of more than 0.7 which are considered foreground or an object. The second category are those that do not overlap a ground-truth object or have less than 0.3 IoU, and are

considered background. A batch of the anchors is sampled, maintaining a balance between foreground and background anchors. This batch is used to calculate the classification loss using binary cross entropy, and the foreground anchors, only, of the batch to calculate regression loss [Ren et al., 2017].

In post processing, Non-Maximum Suppression (NMS) is applied to solve duplicate proposals of one object by discarding the proposals with an IoU above a set threshold with another proposal with a higher score. A too high IoU threshold may lead to too many proposals, and too low can lead to missing proposals [Ren et al., 2017].

5.3.2 Region of Interest Pooling

With the object proposals from the RPN, the next step is to classify the bounding boxes. But before the classification is done, ROI pooling is performed. This is done by extracting the fixed size feature maps for each proposal from the already existing convolutional feature map shared [Ren et al., 2017].

5.3.3 Classification

In the classification step of the network the R-CNN network is employed. With a FC layer as output the R-CNN has two goals; classifying the proposals into classes and to fit the bounding boxes. It flattens the feature map of each proposal and then uses two FC layers of size 4096 with Rectified Linear Unit (ReLU) activation. Then, using two different FC layers for each of the objects, one for classification and one for bounding box regression prediction [Ren et al., 2017]. An example of the flow of a classification by the R-CNN is also shown in Figure 5.7. Here a bicycle is classified and the bounding box is adjusted to fit the object.

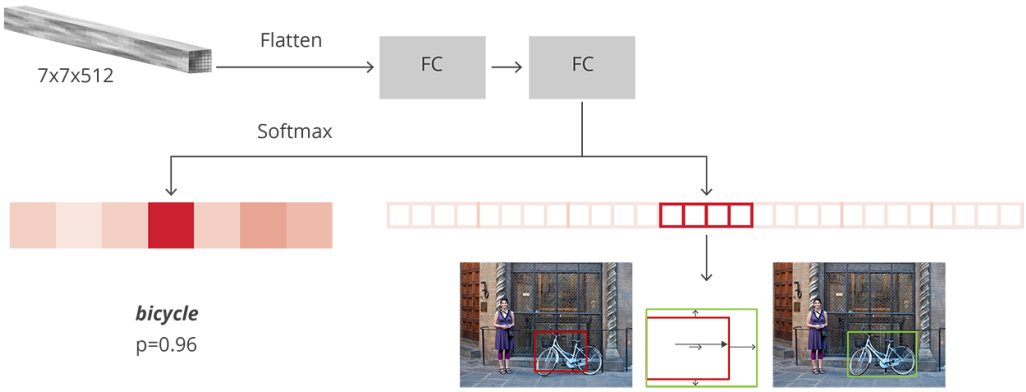


Figure 5.7: Example of classification flow in the R-CNN classifying a bicycle and regressing a bounding box [Rey, 2018]

Chapter 6

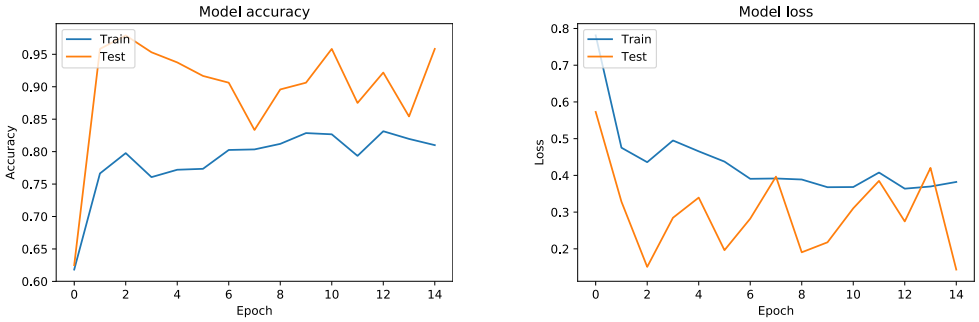
Results

The results presented are both training and testing results from the two solutions. Both the image classification and the multi-class object detection are trained and tested on the 60 seconds video with three zebrafish.

The solutions are trained on annotated data of 411 different occlusions in total and a similar amount annotated occurrences of *no occlusions*.

6.1 Image Classification

The image classification solution is trained for 15 epochs, the results are shown in Figure 6.1



(a) The image classification accuracy for both training and testing

(b) The image classification loss for both training and testing

Figure 6.1: Model accuracy and loss when trained for 15 epochs

As shown in Figure 6.1 the training accuracy is lower than the testing accuracy, however, the training graph is in both functions more stable and seems closer to converging than the testing graph. Meanwhile, the training loss is higher than the test loss as well.

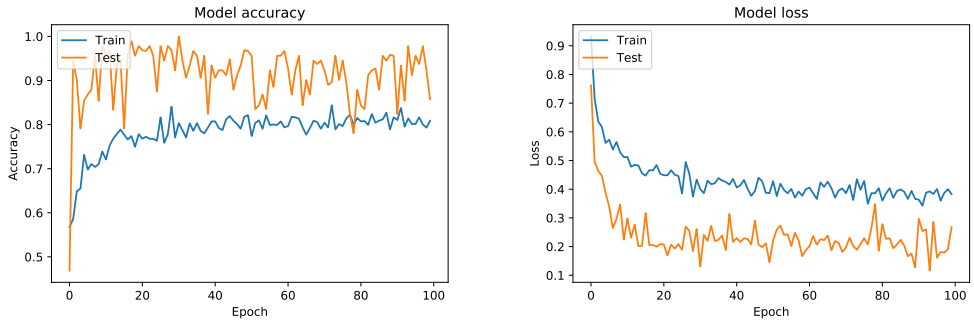
In general, the model is still unstable and needs more training for converging.

Table 6.1 also shows that the test results are better than the training.

Table 6.1: End results achieved with 15 epochs.

Training Loss	0.4841
Training Accuracy	0.7738
Testing Loss	0.2487
Testing Accuracy	0.9318

The image detection model is also trained for 100 epochs, the results of this is shown in Figure 6.2 and Table 6.2.



(a) The image classification accuracy for both training and testing

(b) The image classification loss for both training and testing

Figure 6.2: Model accuracy and loss when trained for 100 epochs

There is still a high amount of instability, but in general the test results are better than the training. It has tendencies towards converging, but it is uncertain due to the instability.

Table 6.2: End results achieved with 100 epochs.

Training Loss	0.3831
Training Accuracy	0.8075
Testing Loss	0.2680
Testing Accuracy	0.8571

6.2 Object Detection

The Faster R-CNN object detection model is trained with a split of 85% for training and 15% for testing.

6.2.1 Training

Training is done at 42 and 152 epochs, as these points had an automatic stop. When training, the best performing weights are saved.

42 Epochs

After the object detection model has been trained for 42 epochs, the training accuracy is as shown in the following figures:

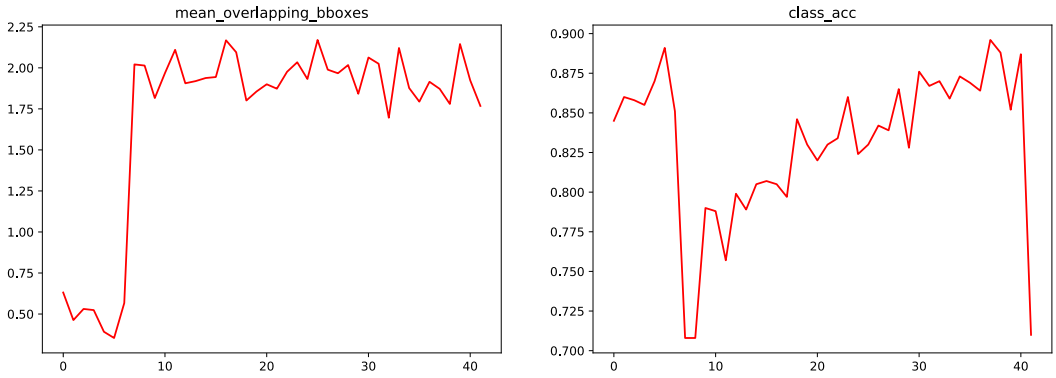


Figure 6.3: Mean of overlapping bounding boxes of ground truth and classification accuracy

Figure 6.3 shows the mean overlapping bounding boxes of the ground truth bounding box, and the classification accuracy of the R-CNN classifier.

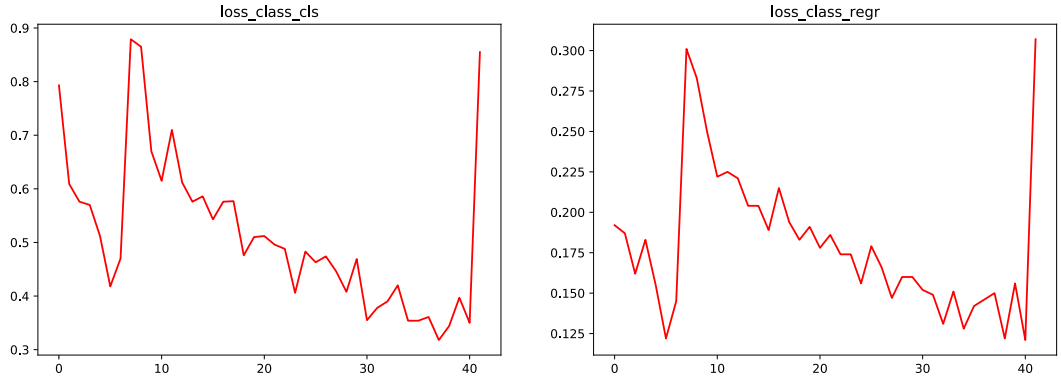


Figure 6.4: Loss in classification and general bounding box regression

Figure 6.4 shows the loss in the R-CNN classifier and bounding box regression.

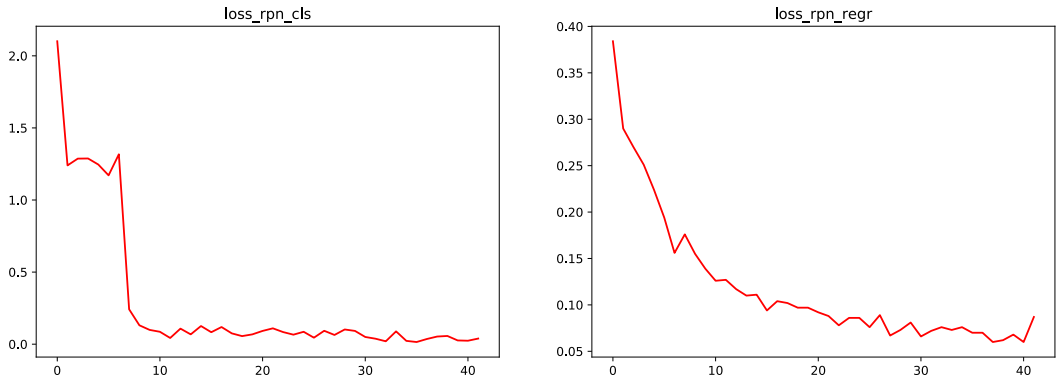


Figure 6.5: Loss in the RPN classification and general bounding box regression

Figure 6.5 shows the loss in the RPN classifier and bounding box regression.

Figure 6.6: Total loss of the entire model

Figure 6.6 shows the total loss of the model during training. The R-CNN classification and regression steps show signs of instability, which is seen in the loss and the accuracy of both overlapping bounding boxes and the classification accuracy.

When testing with these results, the test was stopped due to poor performance.

152 Epochs

After the object detection model has been trained for 152 epochs, the training accuracy is as shown in the following figures:

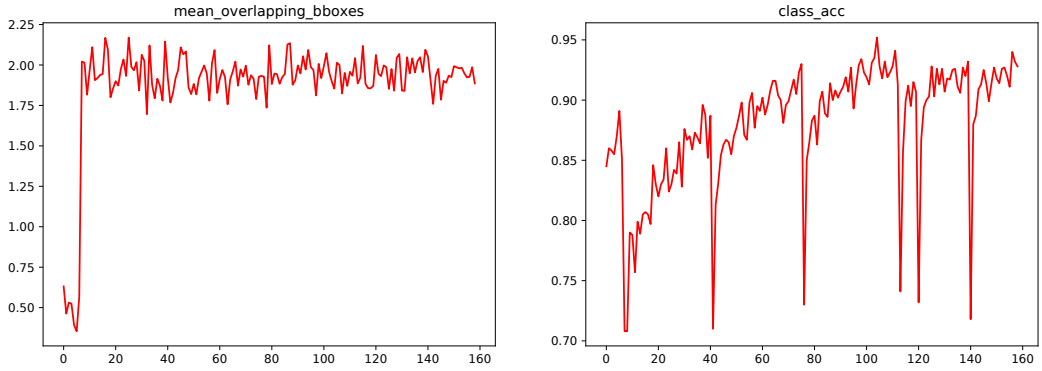


Figure 6.7: Mean of overlapping bounding boxes of ground truth and classification accuracy

Figure 6.7 shows the mean overlapping bounding boxes of the ground truth bounding box, and the classification accuracy of the R-CNN classifier.

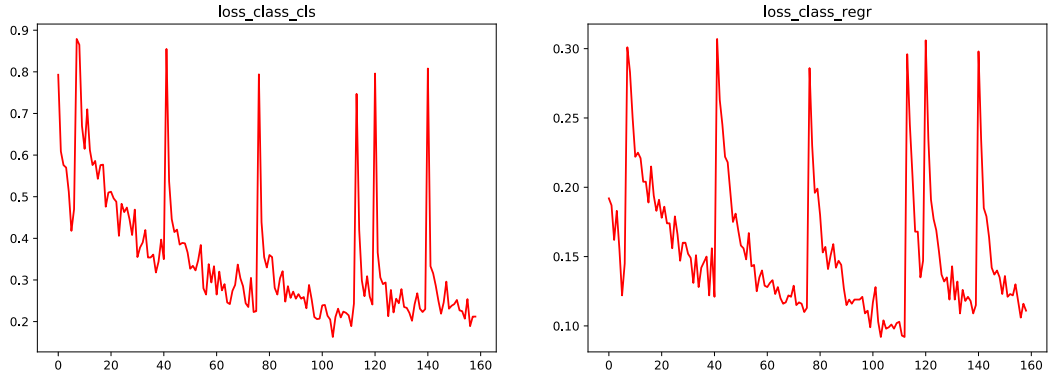


Figure 6.8: Loss in classification and general bounding box regression

Figure 6.8 shows the loss in the R-CNN classifier and bounding box regression.

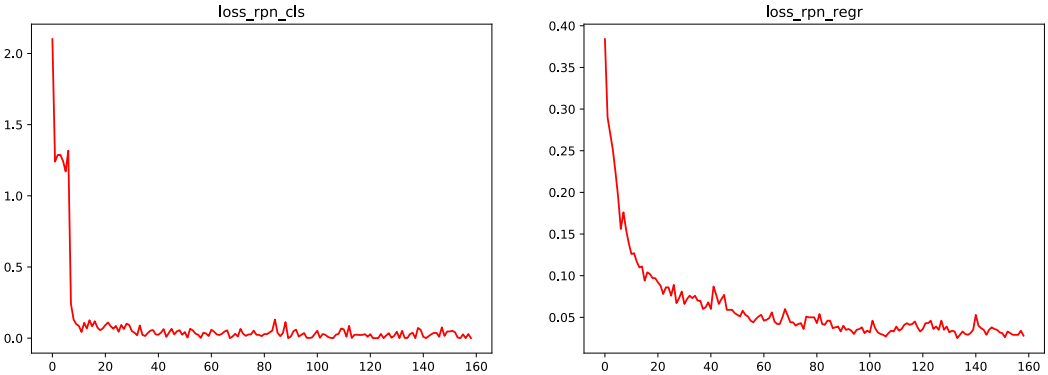


Figure 6.9: Loss in the RPN classification and general bounding box regression

Figure 6.9 shows the loss in the RPN classifier and bounding box regression.

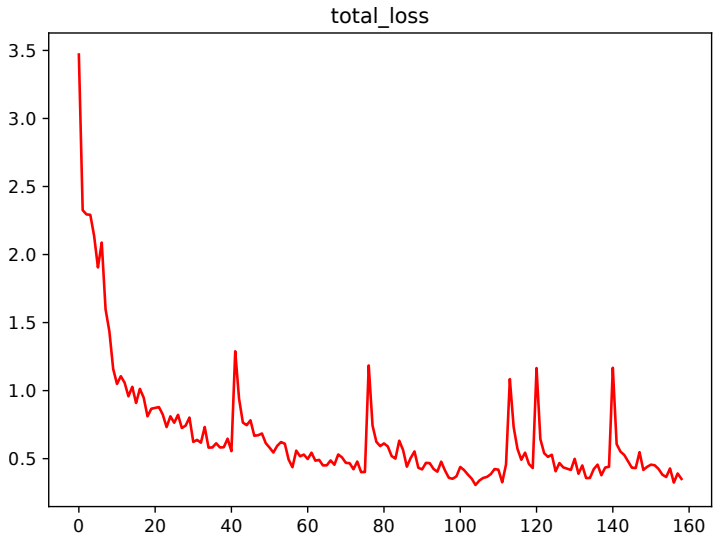


Figure 6.10: Total loss of the entire model

Figure 6.10 shows the total loss of the model during training. There are still signs of instability and again especially in the R-CNN classification and regression. However, there are signs towards the model converging.

6.2.2 Testing

The training and testing split is 85% for training and 15% for testing. With this split the solution reaches a mean Average Precision (mAP) of 66.8%.

Figure 6.11, 6.12, and 6.13 show plots of predictions of three random testing frames, with predictions and bounding box regressions. The predictions and the accuracy is written in the image caption. Only detections with a confidence above 70% are accepted.

Table 6.3 shows that the average precision is very different between classes, ranging from 34.4% to 100%.

Table 6.3: Average precisions of the different occlusion types

Occlusion Type	Average Precision
T Shape	74.6%
V Shape	53.9%
On Top	100%
Cross	78.2%
Elongation	91.9%
Other	37.6%
No Occlusion	34.6%

Figure 6.11 shows two zebrafish creating a T-shape occlusion, which is detected with a confidence of 91.37% and single zebrafish detected as no occlusion with a confidence of 99.48%.

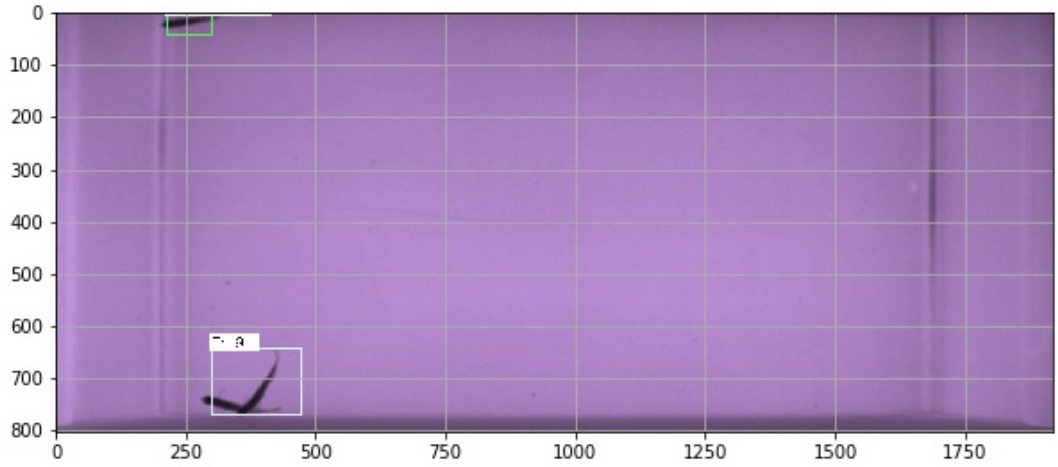


Figure 6.11: Detection of the T-shape is with a confidence of 91.37% and of no occlusion with 99.48%

Figure 6.12 shows a crossing occlusion which is not being detected and a single zebrafish detected as no occlusion with a confidence of 99.43%.

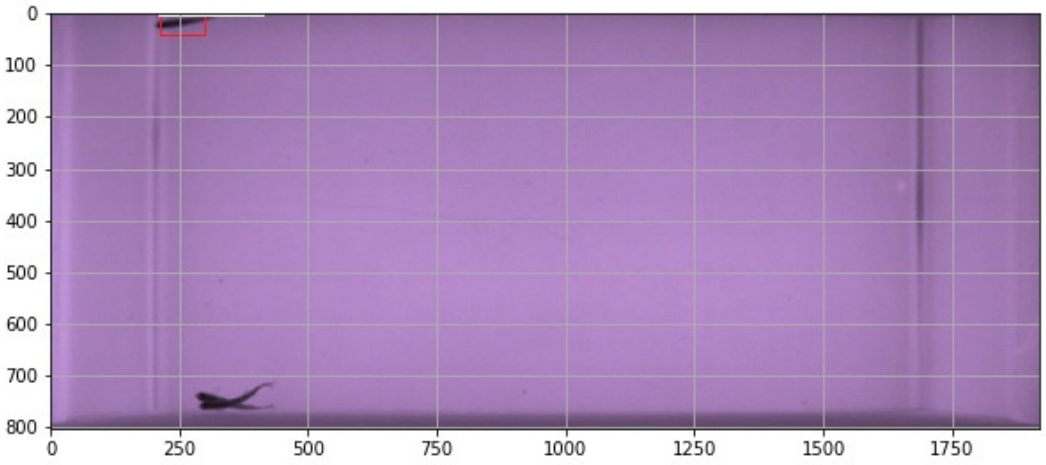


Figure 6.12: Only detection of no occlusion with confidence of 99.43%

Figure 6.13 shows three zebrafish with no occlusions occurring in the image, but only two of them are detected as no occlusions, with confidences of 99.48% and 98.73%. The last zebrafish does not have a detection confidence higher than 70% and is therefore ignored.

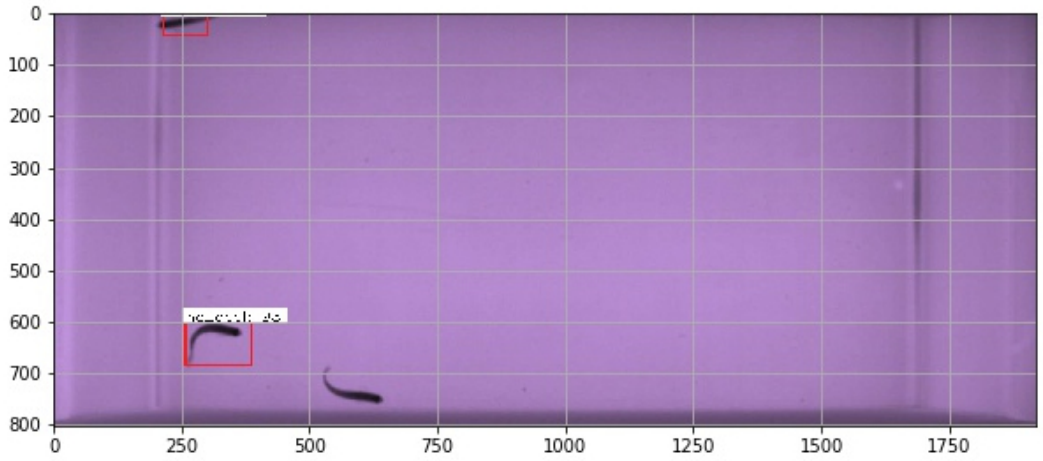


Figure 6.13: Two detections of no occlusions with confidence of 99.48% and 98.73%

Chapter 7

Evaluation

The evaluation is done in regards to the final networks created and how the design and results are able to answer the problem statement and the questions asked in the extension of the problem statement in chapter 3.

In order to be able to answer these questions, data is collected and annotated to be able to train and test the solutions. The data is a 60 seconds long video of three zebrafish in an aquarium. The annotations made are bounding boxes, and are made up of seven different classes; Elongation, On Top, T-shape, V-shape, Cross, Other, and No Occlusion. The "No Occlusions" are also annotated in an effort to let the model detect all occurrences in the aquarium, and being able to separate occlusions and non-occlusions.

According to Dolado et al. [2014] different types of occlusions may require different kinds of computer vision methods and chose to split occlusions into two groups. This report splits the occlusions further into six different categories, in an effort to be able to differentiate between categories as much as possible, should the categories require different methods to separate the zebrafish in the data.

The different categories would only be necessary, if a complete tracking solution aims to solve the occlusions automatically by applying solutions before tracking with methods related to the category. Whereas if a tracking solution would aim to utilise the user, solely being able to detect an occlusion in the image may suffice. In an effort to detect the occlusions in the video, two different solutions has been created.

The first solution is an image classification network, which is trained and tested on 400 images of the two classes each; *occlusions* and *no occlusions*. The results presented are created training for 15 and 100 epochs separately. As the graphs show in Figure 6.2a, testing is especially unstable, as the accuracy fluctuates between 0.75 up to almost 1.0 and without showing any sign of converging. This instability can be caused by multiple issues; as both lack of data, learning-rate, and bad hyper parameters can cause this. An attempt to fix this was to add two dropout layers between the FC layers, but to no avail. Lowering the depth of the FC introduced some stability, but not anything noteworthy.

The second solution is an object detection solution utilising a Faster R-CNN implementation, using the RPN to generate region proposals and VGG16 network for extracting features. With a mAP of 66.8% there is clearly room for improvement. This is also shown in the predictions in testing, neither all occlusions nor the "no occlusions" are detected with a confidence above 70%. In Figure 6.12 the *crossing* occlusion is not detected. This can be due to the shape the two zebrafish make is also close to an *Cross* occlusion.

To improve precision, the first step would be to increase the data volume. As shown in Figure 4.10 there is not even 40 occurrences of the *on top* occlusions in the data, which may be a reason for the model not being able to detect this with a high enough certainty.

The training plots also show some instability, which once again can be caused by the low amount of data.

Chapter 8

Conclusion

Throughout the project, the aim has been to find a solution to the problem statement:
How can a zebrafish occlusion detection solution be implemented in order to potentially optimise zebrafish tracking?

In order to be able to detect zebrafish occlusions two different solutions are made. To be able to cater to different occlusion handling approaches a simple image classification occlusion detection and a more complex multi-class object occlusion detection were developed. The simple image classification would require a user to solve occlusions in a tracking system. While the more complex object detection solution system will enable application of fit solutions based on the category automatically and thereby require no user interaction during tracking.

The image classification achieves a validation accuracy of 93% when trained for 15 epochs, but shows a lot of instability which can indicate a coincidence. When training for longer, no convergence is achieved which may indicate the need of more data or tweaking of the hyper parameters.

The object detection achieves a mean Average Precision (mAP) of 66.8%, and is able to locate both occlusions and single zebrafish as *no occlusion*. It is not able to detect all types of occlusions with some classes having a rather low average precision in testing indicating the need for more data to increase performance.

The report proposes two different solutions on how a zebrafish occlusion detection solution can be made, and presents a novel categorisation of multiple zebrafish occlusions. The multi-class object detection also serves as a proof of concept for the implementation of this solution in a complete tracking system.

Bibliography

Bengtson, S. H. and Pedersen, M., 2017. Tracking Zebrafish in 3D using Stereo Vision.

Dolado, R., Gimeno, E., Beltran, F. S., Quera, V., and Pertusa, J. F., 2014. A method for resolving occlusions when multitracking individuals in a shoal. *Behavior Research Methods*, dec, 47(4), pp. 1032–1043. ISSN 15543528. doi: 10.3758/s13428-014-0520-9. Available at: <http://link.springer.com/10.3758/s13428-014-0520-9>.

Feijó, G. d. O., Sangalli, V. A., da Silva, I. N. L., and Pinho, M. S., 2018. An algorithm to track laboratory zebrafish shoals. *Computers in Biology and Medicine*, may, 96, pp. 79–90. ISSN 18790534. doi: 10.1016/j.compbiomed.2018.01.011. Available at: <https://www-sciencedirect-com.zorac.aub.aau.dk/science/article/pii/S0010482518300192>.

Girshick, R. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pp. 1440–1448, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.169. Available at: <https://github.com/rbgirshick/>.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, nov 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.81. Available at: <http://arxiv.org/abs/1311.2524>.

Green, J., Collins, C., Kyzar, E. J., Pham, M., Roth, A., Gaikwad, S., Cachat, J., Stewart, A. M., Landsman, S., Grieco, F., Tegelenbosch, R., Noldus, L. P., and Kalueff, A. V., 2012. Automated high-throughput neurophenotyping of zebrafish social behavior. *Journal of Neuroscience Methods*, 210(2), pp. 266–271. ISSN 01650270. doi: 10.1016/j.jneumeth.2012.07.017. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2012.07.017>.

Hedges, S. B., nov 2002. *The origin and evolution of model organisms*, Nature Publishing Group. ISSN 14710056. Available at: <http://www.nature.com/articles/nrg929>.

Kalueff, A. V., Gebhardt, M., Stewart, A. M., Cachat, J. M., Brimmer, M., Chawla, J. S., Craddock, C., Kyzar, E. J., Roth, A., Landsman, S., Gaikwad, S., Robinson, K., Baatrup, E., Tierney, K., Shamchuk, A., Norton, W., Miller, N., Nicolson, T., Braubach, O., Gilman, C. P., Pittman, J., Rosemberg, D. B., Gerlai, R., Echevarria, D., Lamb, E., Neuhauss, S. C., Weng, W., Bally-Cuif, L., and Schneider, and the Zebrafish Neuros, H., 2013. Towards a Comprehensive Catalog of Zebrafish Behavior 1.0 and Beyond. *Zebrafish*, 10(1), pp. 70–86. ISSN 1545-8547. doi: 10.1089/zeb.2012.0861. Available at: <www.liebertpub.comhttp://www.liebertpub.com/doi/10.1089/zeb.2012.0861>.

Karpathy, A., 2016. CS231n Convolutional Neural Networks for Visual Recognition. *Stanford University*, pp. 1–22. Available at: <http://cs231n.github.io/>.

Karpova, A. and Haurum, J. B., 2018. Re-Identification of Zebrafish Using Metric Learning, Aalborg.

Ketaro, W., 2016. *labelme: Image Polygonal Annotation with Python*. Available at: <https://github.com/wkentaro/labelme>.

Khan, F. R. and Alhewairini, S. S. Zebrafish (Danio rerio) as a Model Organism. In *Current Trends in Cancer Management [Working Title]*. IntechOpen, nov 2018. doi: 10.5772/intechopen.81517. Available at: <https://www.intechopen.com/online-first/zebrafish-danio-rerio-as-a-model-organism>.

Miller, N. and Gerlai, R., 2012. From Schooling to Shoaling: Patterns of Collective Motion in Zebrafish (Danio rerio). *PLoS ONE*, 7(11). ISSN 19326203. doi: 10.1371/journal.pone.0048865. Available at: <www.plosone.org>.

Perlman, R. L., 2016. Mouse Models of Human Disease: An Evolutionary Perspective. *Evolution, Medicine, and Public Health*, p. eow014. doi: 10.1093/emph/eow014. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4875775/pdf/eow014.pdf>.

Qian, Z. M. and Chen, Y. Q., 2017. Feature point based 3D tracking of multiple fish from multi-view images. *PLoS ONE*, jun, 12(6), p. e0180254. ISSN 19326203. doi: 10.1371/journal.pone.0180254. Available at: <https://dx.plos.org/10.1371/journal.pone.0180254>.

Qian, Z. M., Cheng, X. E., and Chen, Y. Q., 2014. Automatically Detect and track multiple fish swimming in shallow water with frequent occlusion. *PLoS ONE*, sep, 9(9), p. e106506. ISSN 19326203. doi: 10.1371/journal.pone.0106506. Available at: <https://dx.plos.org/10.1371/journal.pone.0106506>.

Ren, S., He, K., Girshick, R., and Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), pp. 1137–1149. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031. Available at: <https://github.com/>.

Rey, J., 2018. Faster R-CNN: Down the rabbit hole of modern object detection. *Tyrolabs*, pp. 1–18. Available at: <https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/>.

Romero-Ferrero, F., Bergomi, M. G., Hinz, R. C., Heras, F. J., and de Polavieja, G. G., 2019. idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. *Nature Methods*, mar, 16(2), pp. 179–182. ISSN 15487105. doi: 10.1038/s41592-018-0295-5. Available at: <http://arxiv.org/abs/1803.04351>.

Stewart, A. M., Grieco, F., Tegelenbosch, R. A., Kyzar, E. J., Nguyen, M., Kaluyeva, A., Song, C., Noldus, L. P., and Kalueff, A. V., 2015. A novel 3D method of locomotor analysis in adult zebrafish: Implications for automated detection of CNS drug-evoked phenotypes. *Journal of Neuroscience Methods*, 255, pp. 66–74. ISSN 1872678X. doi: 10.1016/j.jneumeth.2015.07.023. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2015.07.023>.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W., 2013. Selective search for object recognition. *International Journal of Computer Vision*, sep, 104 (2), pp. 154–171. ISSN 09205691. doi: 10.1007/s11263-013-0620-5. Available at: <http://link.springer.com/10.1007/s11263-013-0620-5>.

Zhang, X., Yang, Y. H., Han, Z., Wang, H., and Gao, C., oct 2013. *Object class detection: A survey*, ACM. ISSN 03600300. Available at: <http://dl.acm.org/citation.cfm?doid=2522968.2522978>.