Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Lotte Eide

Hydrogen Sulfide Detection through Salmon Juvenile Monitoring with Stereo Vision and Machine Learning

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Alver
Co-supervisor: Bjarne Kvæstad, SINTEF Ocean
June 2023

NTNU
Norwegian University of Science and Technology

SINTEF

Lotte Eide

Hydrogen Sulfide Detection through Salmon Juvenile Monitoring with Stereo Vision and Machine Learning

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Alver
Co-supervisor: Bjarne Kvæstad, SINTEF Ocean
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This Master's Thesis was written during the spring of 2023 as a part of the course TTK4900 - Engineering Cybernetics: Master's Thesis and is weighted 30 credits. SINTEF Ocean provided the project in contribution to the research project DigiRAS. The problem formulation includes detecting toxin formation in water by monitoring salmon juveniles utilizing stereo vision and machine learning. This project is an extension of the work conducted during the fall of 2022 within the context of the TTK4550 course - Specialization Project.

A selection of material from the Specialization Project [14] has been restated in this thesis to improve readability and provide context for the reader. The selected material included is listed below.

- From Chapter 1: Parts of Section 1.1.
- From Chapter 2: Parts of Section 2.1, 2.2, 2.3, and 2.4.

I want to thank my supervisor Morten Omholt Alver for valuable feedback and guidance during this project. I also want to thank Bjarne Kvæstad from SINTEF Ocean for technical guidance and helpful advice.

Executive Summary

Background: Expanding land-based production phases in salmon farming is an emerging trend within the aquaculture industry. While reducing time spent at sea can yield many benefits, the development of land-based facilities faces challenges, such as the immaturity of the utilized technology. Currently, the industry lacks autonomous solutions for monitoring fish behavior. One significant challenge of land-based farming is poor water quality, adversely impacting fish welfare. This Master's Thesis explores the integration of stereo vision and machine learning to detect water quality changes, specifically the formation of hydrogen sulfide, by monitoring salmon juvenile behavior. SINTEF Ocean provided the project in contribution to the research project DigiRAS.

Method: A Stereo R-CNN object detection model was trained using a dataset of stereo image pairs featuring salmon juveniles. Camera calibration results were retrieved to recover 3D information about the detected fish in stereo images. The Stereo R-CNN model and camera calibration outcomes were employed in object tracking to extract positional data of salmon juveniles from videos. The videos used contained various scenarios of fish behavior: before, during, and after being exposed to different concentrations of hydrogen sulfide. The acquired results suggested that velocity and speed change rate were the most characteristic indicators of various swimming patterns.

Several datasets were developed based on positional data distributions. An assessment of different techniques for examining the detection of hydrogen sulfide was performed. For testing, several machine learning models were utilized to classify and estimate hydrogen sulfide concentrations to investigate if water quality changes could be detected from the datasets.

Results: The obtained results implied that 3D estimations of positional data could be employed for further analysis. The classification models demonstrated limited success, with a peak accuracy of 56%. Conversely, the regression models provided satisfactory results, signifying that a regression model might be viable for hydrogen sulfide detection. More positional data must be generated in order to assess its credibility.

Sammendrag

Bakgrunn: Utvidelsen av landbaserte produksjonsfaser i lakseoppdrett er en voksende trend innen akvakulturindustrien. Reduksjon av tid tilbrakt på sjøen kan gi mange fordeler, men utviklingen av landbaserte fasiliteter bringer også med seg egne utfordringer, eksempelvis umodenhet av teknologien som blir brukt. En av de største utfordringene forbundet med landbasert oppdrett er dårlig vannkvalitet som påvirker fiskevelferden negativt. For øyeblikket mangler industrien autonome løsninger for å overvåke fiskeatferd. Denne masteroppgaven utforsker bruken av stereosyn og maskinlæring for å oppdage endringer i vannkvalitet, spesielt rettet mot hydrogensulfid, ved å overvåke atferd til laksesmolt. SINTEF Ocean bidro med prosjektet i forbindelse med forskningsprosjektet DigiRAS.

Metode: En Stereo R-CNN objektdeteksjonsmodell ble trent ved bruk av et datasett bestående av stereobildepar som viser laksesmolt. Kamerakalibreringsresultater ble brukt for å gjenopprette 3D-informasjon om fisken som ble detektert i stereobildene. Stereo R-CNN-modellen og resultater fra kamerakalibreringen ble brukt i objektsporing til å generere posisjonsdata for laksesmolt. Videoene som ble brukt inneholdt forskjellige scenarier av fiskeatferd: før, under og etter å ha blitt eksponert for forskjellige doser av hydrogensulfid konsentrasjoner. Resultatene antydet at hastighet og hastighetsendring var de mest karakteristiske indikatorene på ulike svømmemønstre.

Flere forskjellige datasett ble laget basert på fordelinger av posisjonsdata. Forskjellige teknikker ble vurdert for å undersøke muligheten for hydrogensulfid deteksjon. Flere maskinlæringsmodeller ble testet for å vurdere om hydrogensulfid konsentrasjoner kunne bli klassifisert og estimert basert på datasettene.

Resultater: De oppnådde resultatene antydet at 3D-estimater av posisjonsdata kunne brukes for videre analyse. Klassifiseringsmodellene som ble testet oppnådde en nøyaktighet på 56%, som ikke anses å være særlig bra. Til tross for dette ga regresjonsmodellene tilfredsstillende resultater, noe som indikerer at en regresjonsmodell kan være en rimelig tilnærming for oppdagelse av hydrogensulfid. For å vurdere troverdigheten må mer posisjonsdata genereres.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
AutoML	Automatic Machine Learning
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
fps	Frames per Second
GPU	Graphics Processing Unit
H_2S	Hydrogen Sulfide
IoU	Intersection over Union
MAE	Mean Absolute Error
ML	Machine Learning
MOT	Multiple Object Tracking
NN	Neural Network
R-CNN	Region-Based Convolutional Network
RAS	Recirculating Aquaculture System
RBF	Radial Basis Function
ResNet	Residual Network
RMSE	Root Mean Square Error
RoI	Region of Interest
RPN	Region Proposal Network
SORT	Simple Online and Realtime Tracking
SOT	Single Object Tracking
ssh	Secure Shell
SVC	Support Vector Classifier
SVM	Support Vector Machine
SVR	Support Vector Regressor
TN	True Negative
TP	True Positive

1

Introduction

Parts of Section 1.1 are restated from the Project Thesis [14] with modifications and additional details. Subsection 1.1.1 has been restated.

1.1 Motivation

The aquaculture industry is one of the leading industries in Norway, supplying seafood to consumers worldwide. 98.8% of the fish farming industry comprises salmon production, making Norway the world's largest producer of farmed salmon [29]. The early stages of the salmon production cycle take place in land-based facilities. The salmon are nurtured in a controlled freshwater environment until they reach a weight of 100-125 grams before their transfer to seawater cages. In recent years, facilities have been developed to grow the smolt up to 1,000 grams [30].

Land-based farming brings several advantages over its offshore counterpart. Fish raised in these controlled environments are shielded from sea lice and other diseases typically associated with offshore farming. Land-based facilities also offer environmental advantages as they have the ability to gather generated waste and other potentially harmful byproducts. Additionally, it offers fish production closer to the local market and in land-locked countries. Over the past two decades, Norway has developed land-based Recirculating Aquaculture Systems (RAS) as a substitute for the traditional flow-through system. RAS recycle the water instead of replacing it, making it less dependent on access to fresh water [3].

Although land-based fish farming offers many benefits, it introduces some new challenges. The biological conditions in RAS can be hard to control and lead to poor water quality. A notable concern is the production of hydrogen sulfide (H_2S), a highly toxic gas that can cause mass mortality in fish populations. At present, there are no automated methods for detecting this toxic gas in water [33]. However, alterations in fish behavior have been noted upon H_2S exposure. While experienced fish farmers can recognize these behavioral changes, an automated system would provide a more objective and consistent monitoring solution. Such a system would not rely on a specific individual's presence and would be capable of continuous operation. Hence, the goal is to create an automated system capable of detecting H_2S exposure by monitoring behavioral changes in salmon, thereby safeguarding the fish's health and the industry's productivity.

1.1.1 DigiRAS

The DigiRAS project is a European research initiative addressing challenges related to land-based fish farming. SINTEF Ocean is one of eleven partners of the project and is leading many of its tasks. The project aims to help RAS facilities operate more efficiently with digitalization [12].

1.2 Problem Description

One of SINTEF Ocean's tasks for the DigiRAS project is to develop a computer vision system to monitor salmon juveniles' behavior to detect H_2S. SINTEF Ocean has conducted an experiment where concentrations of H_2S in a salmon juvenile RAS tank were increased. They gathered video footage from the tank during the experiment from a stereo vision setup.

In the fall of 2022, an object-tracking system was established during the Specialization Project in collaboration with the previous work of Åshild Nedreberg [31]. This system utilized Stereo R-CNN to detect objects, a SORT algorithm to track fish, and a Kalman filter to estimate their 3D positions. Initial tests have indicated that velocity distributions retrieved from object-tracking effectively differentiate between two distinct behavioral patterns of fish: calm and stressed. This project aimed to develop this system further.

The objectives of this Master's Thesis are summarized below.

- Further develop the existing annotation tool for video data and establish a

more extensive training dataset for the Stereo R-CNN model.

- Further develop the tracking algorithm to obtain longer and more precise fish position and velocity time series.
- Analyze larger parts of the video material to build a more comprehensive dataset.
- Test one or more methods for detecting various stress levels based on the H_2S experiment.
- Assess how well the stress level can be estimated and recommend future work.

1.3 Delimitations

The scope of this study is limited to the analysis of an existing object tracking module consisting of a Stereo R-CNN, a SORT algorithm, and a Kalman filter for 3D estimation and tracking of salmon juveniles. Other methods for object tracking have not been considered.

Further, one of the main objectives of this Master's Thesis was to investigate and assess one or more methods for detecting various stress levels in the context of the H_2S experiment. Rather than attempting to classify specific fish behaviors, the focus shifted towards detecting H_2S concentration utilizing positional data.

1.4 Structure of the Report

This report is divided into six chapters. Chapter 1 presents the motivation and problem description for the project.

Chapter 2 presents the background theory for this project, divided into five main sections: recirculating aquaculture systems, the H_2S experiment, stereo vision, machine learning, and object tracking.

Chapter 3 presents the materials and methods used to complete this project. This mainly includes hardware and software, preparing image data for Stereo R-CNN, object tracking, processing acquired positional data, and classification and estimation of H_2S concentration.

Chapter 4 presents the obtained results from this work. The results are discussed and evaluated in chapter 5, and suggested future work is provided. Chapter 6 summarizes the discussion as a conclusion.

2

Theory

This chapter presents the background theory of the master project. Section 2.1 and 2.2 are partly restated from the Project Thesis [14]. Subsection 2.1.1 is refined to include more information on the topic. Section 2.3 is restated with alternations in Subsection 2.3.1. Parts of Section 2.4 have been restated, including parts of Subsection 2.4.1 and 2.4.6.

2.1 Recirculating Aquaculture System

Recirculating Aquaculture System (RAS) is a land-based technology for farming fish or other aquatic organisms by reusing the water in production. The technology is based on mechanical and biological filters and is designed to minimize water consumption, control culture conditions, and allow waste streams to be fully managed. The use of RAS compared to traditional flow-through systems proliferates in many areas of the fish farming sector [5].

RAS has excelled in terms of environmental impact. The limited amount of water used is beneficial as water has become a limited resource in many regions. RAS can recycle up to 90-97% of the freshwater used in land-based farming. This makes up 0.1%-1% of the water consumed in a traditional flow-through system [19]. The production is thus more independent of external conditions and provides a more stable environment for the fish.

Although RAS has many beneficial aspects, some challenges remain. In particular,

the production of hydrogen sulfide.

2.1.1 Hydrogen Sulfide in RAS and its Effects

Hydrogen Sulfide (H_2S) is a colorless toxic gas to humans and fish and has a strong unpleasant odor. The presence of H_2S is one of the most significant challenges in RAS, posing a threat to the health and welfare of the fish.

There are numerous factors contributing to the generation of H_2S in RAS. It is produced as a byproduct when certain bacteria decompose organic matter, such as fish waste or unconsumed feed. Conditions promoting H_2S production include stagnant water and sludge, dead zones or corners, thick biofilters, and sediment buildup within piping systems [36]. These elements can adversely effect water quality and fish health in aquaculture environments.

H_2S concentrations of $2\mu g/L$ in freshwater and $5\mu g/L$ in salt water can cause stress for fish, and concentrations above $25\mu g/L$ can be lethal. In the case of H_2S poisoning, the fish gills are damaged, impairing their ability to extract oxygen from the water and leading to suffocation. In addition, H_2S can cause tissue damage and organ failure, leading to illness and death [35].

These adverse effects can make the fish break from its regular swimming pattern. The general characteristics of salmons' regular swimming pattern are consistent and steady swimming, and maintaining their position in the water column without signs of disorientation. In case of H_2S exposure, the fish can exhibit various indicators of stress and discomfort, and some common signs include [2]:

- **Rapid or labored breathing**: The symptoms are increased gill movement, gasping, or rapid breathing if the H_2S inhibits the fish's ability to breathe.
- **Erratic swimming or loss of equilibrium**: Fish affected by H_2S may exhibit abnormal swimming behavior, such as erratic movements, disorientation or loss of balance, change of tail-beating frequencies, swimming sideways, swimming in circles, or bolting around. In severe cases, they may lose their ability to maintain their position in the water column and sink or float [24].
- **Lethargy**: Fish exposed to H_2S may become less active, sluggish, or show reduced responsiveness to stimuli.
- **Aggregation near the water surface or water inlets**: As the H_2S is more concentrated in low-oxygen environments, fish may seek areas with higher

oxygen, such as near the water surface or close to water inlets where fresh water is being added to the system.

- **Darkening of color**: Fish changing their coloration, often darker than usual, is usually caused by physiological stress.
- **Reduced appetite or feeding**: Fish under stress from H_2S exposure may show reduced appetite and lower activity during feeding periods.
- **Mortality**: In severe cases of H_2S exposure, fish may die due to the toxic effects on their respiratory and physiological systems.

2.2 Hydrogen Sulfide Experiment

SINTEF Ocean conducted an H_2S experiment in contribution to the DigiRAS project. The experiment started on 27th of June 2022 and ended on 7th of July 2022. The setup consisted of three RAS tanks, where two were gradually exposed to H_2S during the experiment period. One of the tanks contained 70 salmon juveniles with a mono and a stereo camera setup. The second tank contained 200 salmon juveniles and a mono camera. These tanks were connected to a water reservoir via water pipes. H_2S was added to the reservoir, leading to H_2S exposure in both tanks. During the experiment, the reservoir was equipped with an instrument that measured salinity, temperature, oxygen, carbon dioxide, and H_2S. The third tank was not connected to the reservoir and was employed for control purposes. An illustration of the experimental setup and procedure is shown in Figure 2.1. The experiment protocol is summarized in Table 2.1.

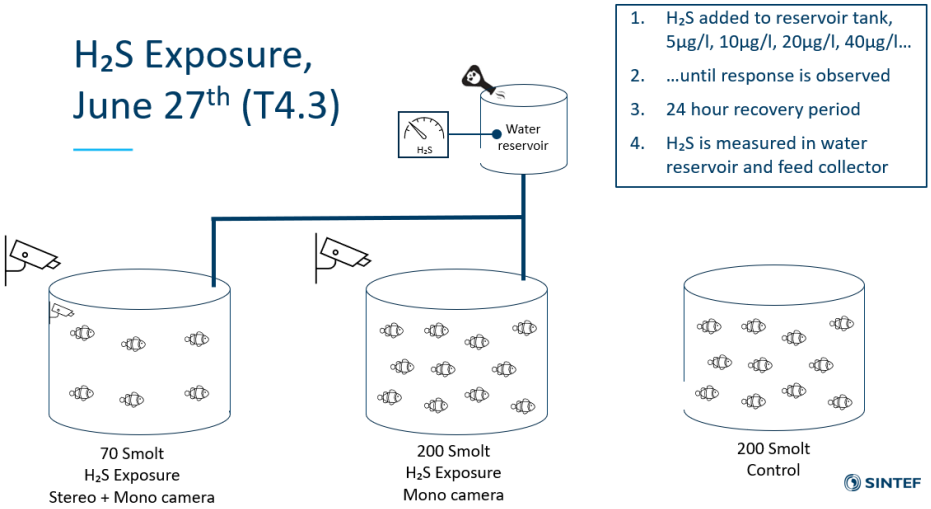


Figure 2.1: SINTEF Ocean's H_2S experiment setup[1].

Table 2.1: SINTEF Ocean's H_2S experiment protocol.

H_2S **experiment protocol**	
1.	Gradually add H_2S to the reservoir tank over a time period every day. The amount varied between $5\mu g/L$ on the first days and $160\mu g/L$ on the last days. The amount was added in 1 liter of water in rounds per minute...
2.	... until the response of the fish was observed.
3.	After the exposure to H_2S, the fish would have a 24-hour recovery phase.
4.	H_2S is measured in the water reservoir and feed collector.

[1]Figure provided by SINTEF Ocean.

2.3 Stereo Vision

Stereo vision is a machine vision technique that uses multiple cameras to perceive depth and create 3D scene models. The cameras are typically arranged side by side with a small distance between them to capture slightly different perspectives of the same scene. By comparing the images from the cameras and calculating the difference in the position of objects in the scene, the machine can determine the distance to each object and create a 3D model of the environment.

A stereo vision system is based upon the human visual system, which consists of two frontal-parallel eyes using binocular vision to see the 3D world. In comparison, stereo vision usually consists of two cameras, illustrated in Figure 2.2. The images captured by these cameras are used to recover surface geometry and distance information of the 3D world from 2D images [20].

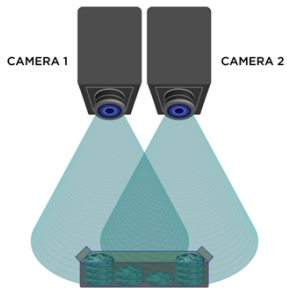


Figure 2.2: Stereo vision setup[2].

2.3.1 Stereo Vision Principle

A stereo vision setup provides image pairs from left and right cameras, called stereo images. The essential requirements for reconstructing a 3D point from a stereo image are *camera parameters* and *disparity estimation*.

Camera Calibration

Camera calibration is the process of finding the parameters of a camera model. It helps establish the relationship between 2D points in an image and the 3D world. These parameters include both the *intrinsic* parameters (focal length and optical

[2]Figure adapted from [8].

center), which are specific to the camera, and *extrinsic* parameters (rotation and translation), which describe the camera's position in the world. For a more detailed elaboration on these parameters and stereo camera calibration, see Subsection 2.3.2 in Project Thesis [14].

Epipolar Geometry and Point Correspondence

Epipolar geometry describes the geometric relationship of a stereo vision setup. This relationship is shaped by various factors, including the cameras' intrinsic and extrinsic parameters. It is represented by a 3×3 matrix known as the *essential matrix* for calibrated cameras and the *fundamental matrix* for uncalibrated cameras.

The cameras of a parallel stereo vision setup have parallel optical axes and are separated by a baseline distance in the x-direction. This geometry is illustrated in Figure 2.3. An object point (P) in the scene will be captured from two distinct perspectives by the left and right camera, $\mathrm{O_L}$ and $\mathrm{O_R}$. The perspective projections of this point in the left and right views, $\mathrm{P_L}$ and $\mathrm{P_R}$, form a conjugate pair. The plane that passes through the left and right camera center and the object point is called the *epipolar plane*, and the intersection of this plane with the image plane is called the *epipolar line*.

In a parallel stereo vision setup, all the epipolar lines are parallel and have the same vertical coordinates in the left and right image planes. This alignment is achieved using the essential matrix, which allows for *image rectification*. The rectification process horizontally aligns the left and right images, resulting in the epipolar lines running parallel to the x-axis. This configuration simplifies finding corresponding points in the left and right views, as they must lie on the same epipolar line. Consequently, a one-dimensional search is sufficient to find the corresponding point of a projection.

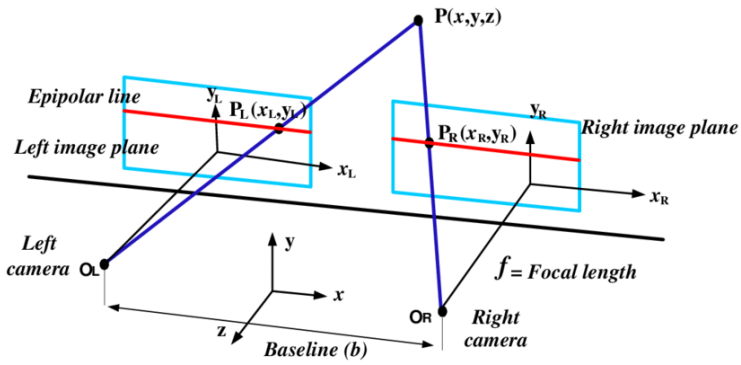


Figure 2.3: The epipolar geometry of a parallel stereo vision setup: two cameras, which view the same scene, detecting a common 3D point P on different 2D locations[3].

Disparity Estimation

Disparity estimation is the process of finding the pixels in the left and right image that corresponds to the same 3D point in the scene. The corresponding points will have different locations in the images since they are captured from slightly different points of view. The epipolar geometry of the setup is utilized to reduce the search space for finding these corresponding points. When the corresponding points have been found, it is possible to determine the distance between them, referred to as *disparity*. The disparity is given by $d = x_L - x_R$, where x_L and x_R are the x-coordinates of the conjugate pair in the left and right image plane.

Once the disparity is found, it becomes feasible to determine the point's depth, z. This is achieved through the comparison of two similar triangles formed by the system's geometry, expressed as:

$$\frac{x_L}{f} = \frac{x + \frac{b}{2}}{z} \tag{2.1}$$

$$\frac{x_R}{f} = \frac{x - \frac{b}{2}}{z} \tag{2.2}$$

x is the x-coordinate of the point in the world scene and b is the baseline distance. f signifies the focal length, which is the distance between the image plane and the optical center of the camera lens. By subtracting Equation 2.2 from Equation 2.1, the calculation of depth is derived [1]:

[3]Figure adapted from [1].

$$z = \frac{b \times f}{x_L - x_R} = \frac{b \times f}{d} \tag{2.3}$$

2.4 Machine Learning

Machine Learning (ML) is a field within artificial intelligence that provides models to make predictions, estimations, and classifications based on historical data. There are two basic approaches within ML: *supervised learning* and *unsupervised learning*. In supervised learning, the model is trained on labeled data to classify or predict outcomes. Unsupervised learning trains the model on unlabeled data, allowing it to find patterns and relationships within the dataset [9].

Parametric vs. Non-Parametric Models

ML algorithms can be broadly classified into *parametric* and *non-parametric* models. Parametric models require the specification of some parameters in advance of making predictions. In contrast, non-parametric methods do not rely on predetermined parameter values, allowing them to deliver more precise outcomes in many instances, as the parameters can be hard to determine [32].

Training, Validation, and Test Dataset

A common practice in supervised ML is to split the available data into three sets: *training*, *validation*, and *test set*. The training set is utilized for training the model, the validation set is used to evaluate the model's performance and tune its hyperparameters, and the test set is used to evaluate its final performance. This allows for a more thorough evaluation of a ML model [18].

Data Augmentation

Data augmentation is a technique used to artificially increase the size of a dataset by creating new data samples based on existing ones. This is often done in the context of training ML models, where having more data can improve model performance. There are many ways to perform data augmentation. Some standard techniques include adding noise to existing data samples and combining multiple data samples to create new ones. Data augmentation can help prevent overfitting

and improve the generalization of a model, as it provides the model with more diverse and varied data to learn from [13].

2.4.1 Support Vector Machine

A Support Vector Machine (SVM) is a supervised ML algorithm that can be used for classification, regression, and outlier detection. It is considered a non-parametric model despite having a few tunable parameters, as it does not make strong assumptions about the underlying structure or distribution of the data. Compared to other ML methods, SVM is powerful at recognizing subtle patterns in complex datasets [22].

Support Vector Classification

A Support Vector Classifier (SVC) aims to create a line or hyperplane that can linearly separate a dataset into classes. Once the hyperplane has been found, new data can easily be classified by determining which side of the hyperplane it falls on.

Consider a binary classification given a training set $\{(x_1, y_1)...(x_n, y_n)\}$ with $y \in \{\pm 1\}$, the decision boundary parameterized by (w, x) can be defined as:

$$\begin{aligned} \langle w, x_i \rangle + b \geq 0 \quad \text{for} \quad y_i = +1 \\ \langle w, x_i \rangle + b < 0 \quad \text{for} \quad y_i = -1 \end{aligned}$$

where $\langle , \rangle$ denotes the dot product of w and x_i. This boundary aims to maximize the *margin*, the distance from the hyperplane to the closest data point in either class, called *support vectors*. An illustration of the SVC concept and its definitions is represented in Figure 2.4.

The maximization of the margin can be written as a linear constrained convex optimization problem, with w as a weight vector:

$$\min_{w,b} \quad \frac{1}{2}||w||^2 \tag{2.4}$$

$$\text{s.t.} \quad y_i(\langle w, x_i \rangle + b) \geq 1 \qquad \forall \ i. \tag{2.5}$$

[4]Figure adapted from [28].

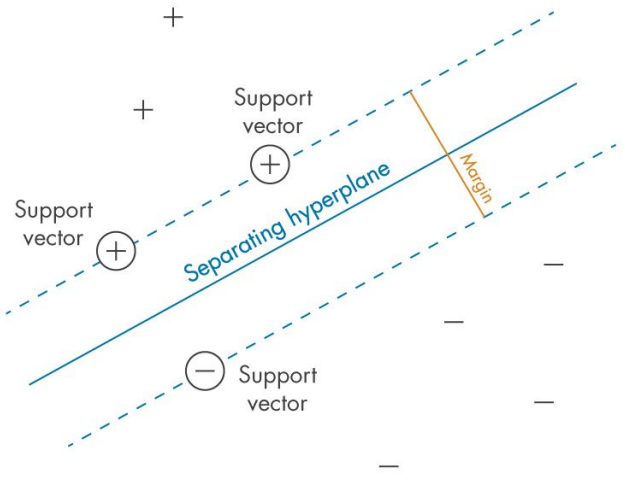


Figure 2.4: SVC concept: Defining the margin between classes - the criterion SVCs seek to optimize[4].

This objective function is a *quadratic program* (QP) with a single global minimum. The optimization problem assumes that the data is linearly separable. In the case of non-linearly separable data, the linear constraints, Equation 2.5, will not be satisfied. A method of dealing with such datasets is loosening the constraints in order to learn a useful classifier. This approach is called *soft margin* and introduces *slack variables*, ξ_i, to allow certain constraints to be violated. Each training point gets a corresponding slack variable. This allows for some training points to be within the margin and be wrongly classified. Mathematically, by introducing slack variables to Equation 2.4, a new optimization problem can be defined as:

$$\min_{w,b,\xi} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \tag{2.6}$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \qquad \forall\ i \\ & \xi_i \geq 0 \qquad \forall\ i \end{aligned} \tag{2.7}$$

where C is the margin parameter, and ξ_i $(i = 1, ..., m)$ are the non-negative slack variables. The C parameter defines the tradeoff between the margin maximization and the classification error minimization. It works as a regularization term and penalizes large values of ξ_i. Large values of C will lead the optimization problem to choose a smaller-margin hyperplane to fit all the training points, potentially

leading to overfitting. Controversially, small values for C will cause the optimizer to choose a larger margin, even if it leads to misclassifying more points.

Support Vector Regression

The SVM was adapted to support regression tasks by introducing Support Vector Regression (SVR). In SVR, the goal is to find the hyperplane that holds the maximum training observations within the margin ϵ. Given a training set $\{(x_1, y_1)...(x_n, y_n)\}$ with $x_i \in \{\mathbb{R}^d\}$ and $y_i \in \{\mathbb{R}\}$ where d is the dimension of samples, the nonlinear function between the input and the output is formulated as:

$$y = f(x) = w^T x + b,$$

where $w \in F$ is the vector of weight coefficients, and b is a bias constant. The w and b are estimated by minimizing the optimization problem:

$$\min_{w,b} \quad \frac{1}{2}||w||^2 \tag{2.8}$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \leq \epsilon \qquad \forall\ i \\ & y_i(\langle w, x_i \rangle + b) \geq -\epsilon \qquad \forall\ i \end{aligned} \tag{2.9}$$

Similar to SVC, slack variables, ξ and ξ^*, are introduced to penalize points from ϵ-insensitive band [17]:

$$\min_{w,b,\xi} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \tag{2.10}$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\langle w, x_i \rangle + b) \leq \epsilon + \xi_i \qquad \forall\ i \\ & y_i(\langle w, x_i \rangle + b) \geq -\epsilon - \xi_i^* \qquad \forall\ i \\ & \xi_i, \xi_i^* \geq 0 \qquad \forall\ i \end{aligned} \tag{2.11}$$

Kernel Functions

All SVMs use the *kernel trick* to bridge linearity and non-linearity. The key idea of this concept is to map the input data from the feature space into higher-dimensional space using a *kernel function*. By mapping the data, it might become linearly separable, and an optimal hyperplane can be found in this space [34].

The *linear kernel* is the simplest and most computationally efficient kernel. It is commonly used for linearly separable datasets with many features. The kernel function calculates the distance between the data points and the hyperplane of the SVM and is denoted:

$$K(x_i, x_j) = \langle x_i, x_j \rangle,$$

where x_i, x_j are input vectors. x_i represents the data points and x_j represents the weights of the hyperplane. As the linear kernel only performs dot product between samples, which is the same as using the original features of the samples, no transformation is applied. Therefore, the linear kernel is often referred to as the baseline kernel to compare the performance of more complex kernels.

The *Radial Basis Function (RBF) kernel*, also known as the *Gaussian kernel*, is a nonlinear kernel function that can capture complex relationships in the data. For large datasets, it can be computationally expensive. It is denoted:

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2},$$

where $||x_i - x_j||^2$ is the squared Euclidean distance between the two feature vectors x_i, x_j. *Gamma*, γ, is a parameter that controls the width of the Gaussian function. Intuitively, it determines how much influence each data point has on the decision boundary. A large value of gamma means that each data point has a greater influence on the boundary. Too large values will lead to overfitting, and no values of C can prevent it. A small value of gamma means that each data point has a smaller influence on the decision boundary. Underfitting can occur when the gamma is too small, making the model too constrained to capture the complexity of the data [25].

Polynomial kernel is another common nonlinear kernel function and is defined as:

$$K(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + c_0)^d,$$

where x_i and x_j are input vectors, c_0 is a constant added to the input vectors' dot product, and d is the polynomial degree. The degree of the polynomial controls the complexity of the decision boundary — higher degree results in higher complexity and vice versa [34].

2.4.2 Decision Tree

A decision tree is a non-parametric supervised learning algorithm utilized for classification and regression tasks [32]. The model provides an inference that is easily understood. The structure of a decision tree is an upside-down tree, beginning with a *root node* at the top, branching out via *edges*, through *decision nodes*, to *leaf nodes*. Figure 2.5 visualizes the concept.

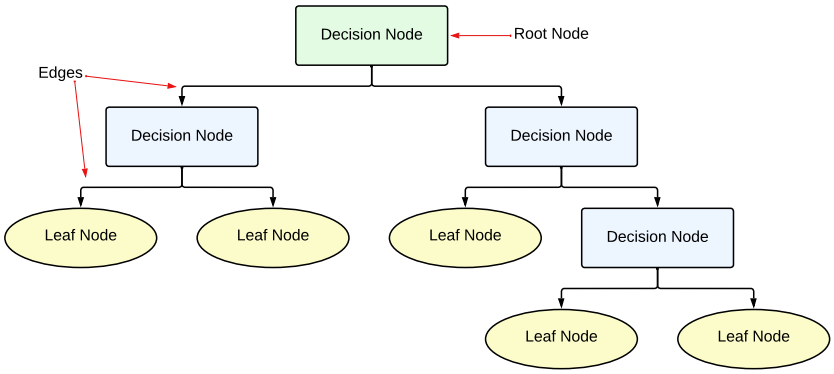


Figure 2.5: Decision tree structure.

The goal is to find the optimal decision tree by minimizing the generalization error. However, it has been established that finding a minimal decision tree is NP-hard and only feasible for minor problems. Heuristic methods tackle larger and more complex datasets, offering practical solutions while maintaining reasonable accuracy. These heuristic methods predominantly fall into two categories: *top-down* and *bottom-up* approaches. Among these, the top-down methods are more commonly used due to their intuitive nature and ease of implementation.

One such top-down decision tree inducer is the Classification and Regression Tree (CART) algorithm. This approach constructs binary trees; each decision node has two outgoing edges. It iteratively subdivides the data into more homogeneous subsets, starting from the root node and proceeding downwards. This process continues until a *stopping criterion* is met, resulting in a decision tree that best

approximates the underlying data distribution. Common stopping criteria include *max depth* and *minimum number of instances per leaf*. The max depth parameter determines the maximum depth of the tree, while the minimum number of instances per leaf stops the tree's growth when the number of instances in a leaf node is below a specified threshold.

A notable characteristic of CART is its capability to construct regression trees. Unlike conventional decision trees that predict classes, regression trees predict real numbers at their leaf nodes. When dealing with regression tasks, CART searches for splits that minimize the squared prediction error, also known as the least-squared deviation. The predicted value at each leaf node is calculated as the weighted mean for that particular node [32].

2.4.3 Ensemble Methods

Ensemble methods are a class of ML techniques that aim to improve the performance of a model by combining multiple individual models, often referred to as base learners or weak learners. The idea behind ensemble methods is to train multiple models with different parameters and combine their outputs to predict the outcome. The combination of individual predictions often has better overall accuracy than one prediction alone [6].

2.4.4 Random Forest

Random forest is an ensemble method used for classification and regression tasks. It is constructed by a collection of decision trees with controlled variation. Each tree in the random forest is constructed independently from a random sample with replacement from the training data. Statistically, the sample will contain 64% of the training set, called *in-bag* instances. The remaining 36% are referred to as *out-of-bag* instances. Each tree in the random forest acts as a base predictor to determine its label. The final output label for classification tasks is typically decided via *majority voting*, where each decision tree casts one vote for its predicted label, and the label with the most votes is used to predict the outcome. In the case of regression tasks, it is common practice to calculate the average of all output values [16].

2.4.5 Deep Learning

Deep learning is a subset of ML that involves using artificial *neural networks* with multiple layers of computation to model complex patterns in data. Deep learning has achieved state-of-the-art performance on various tasks, including image recognition, natural language processing, and speech recognition [15].

Neural Networks

Neural Networks (NN) are the fundamental building block of deep learning. A NN is inspired by the structure and function of the human brain and simulates this through algorithms. NNs consist of three layers: an input layer, one or more hidden layers, and an output layer. Each layer is composed of units, the simplest function of deep learning, which can be associated with neurons in the brain. Each unit receives input from other units in the previous layer, applies a mathematical transformation to the input, and then sends the result to the units in the next layer [15].

A unit has a set of associated *learning parameters*, *weights*, and an *activation function*. The learning parameters are hyperparameters that control the training process itself. The weights of the connections are adjusted during the training process, allowing the NN to learn complex patterns in the data. The activation function decides if the unit can pass information or not. If the NN has more than one hidden layer, it is called a deep neural network [15].

2.4.6 Stereo Region-Based Convolutional Neural Network

A Stereo Region-Based Convolutional Neural Network (Stereo R-CNN) is a deep learning model for stereo vision tasks, such as depth estimation and 3D object detection. It was first proposed by P. Li et al. in 2019 to serve visual perception, motion prediction, and planning for autonomous driving [26]. It takes a stereo image pair captured in the same time frame and detects objects using stereo matching. An illustration of a Stereo R-CNN architecture is provided in Figure 2.6.

Stereo R-CNN Architecture

Stereo R-CNN is based on Faster R-CNN architecture. The Faster R-CNN is designed to be faster and more efficient than the original R-CNN, substituting the

time-consuming feature extraction method, *Selective Search*, with a CNN feature extractor. For the Stereo R-CNN in Figure 2.6, this feature extraction is done by a pre-trained *Residual Network* consisting of 101 layers (ResNet-101) and a *Feature Pyramid Network* (FPN). Further, the Stereo R-CNN architecture can be divided into three main components: a *stereo Region Proposal Network* (RPN), a *stereo regression* module, and a *keypoint prediction* module [26].

The stereo RPN is an RPN that is modified to work with stereo images. It generates region proposals for potential objects in the scene based on the feature maps from the FPN [27]. The region proposals are fed into *Region of Interest Align* (RoI Align), where the extracted features are aligned with the region proposals [21].

After applying RoI Align on the left and right feature maps, the left-right features are linked and fed into a stereo regression model. This module generates a 3D representation of the objects in the scene. An object's 3D representation includes its 3D location, classification, and a 2D bounding box representing its size.

Simultaneously in a *keypoint prediction* branch, the location of a set of keypoints is predicted. Each object has a set of keypoints representing the object's borders. The keypoints and the output of the stereo regression module are further fed into a 3D box estimation module. The 2D bounding boxes are combined with the keypoints to create a 3D bounding box. Lastly, the output is sent to a 3D box alignment module to refine the 3D localization of the objects further. The final output of the Stereo R-CNN is a set of 3D bounding boxes wrapping the detected objects and their respective classes [26].

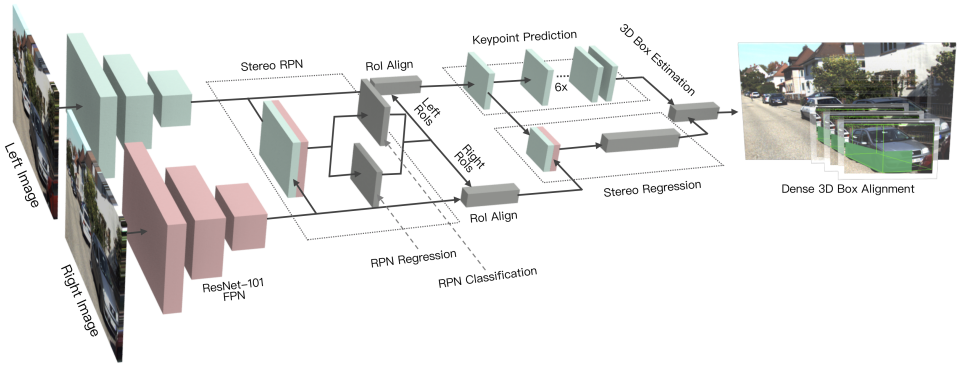


Figure 2.6: Stereo R-CNN architecture[5].

[5]Figure adapted from https://github/Stereo-RCNN.

2.4.7 Automatic Machine Learning Methods

Automatic Machine Learning (AutoML) represents an innovative methodology aimed at automating the end-to-end process of applying ML to resolve different tasks. It utilizes techniques and methods to simplify the process of model selection, hyperparameter tuning, iterative modeling, and model evaluation in ML. This approach allows newcomers to enter the ML field and apply ML techniques for data analysis.

The key elements of AutoML include [23]:

- **Model selection**: One of AutoML's key features is automatically selecting the best model for a given task. It evaluates various models for the given problem and selects the one with the best performance.
- **Hyperparameter tuning**: ML models often have numerous hyperparameters that must be set before training begins. The manual tuning process can be time-consuming and difficult. AutoML utilizes techniques like grid search, random search, or Bayesian optimization to automatically tune these hyperparameters, sparing the user of the tuning process.
- **Feature engineering**: AutoML includes automated feature engineering. This is the process of automatically creating new features from the input data that might improve the model's performance.
- **Model evaluation**: AutoML systems automatically evaluate the performance of the models using predefined metrics. This includes cross-validation or bootstrapping techniques to estimate the model's generalization error.
- **Model interpretability**: Some AutoML systems include tools to interpret the model, providing insights about the importance of features and how they are combined to make estimations.

Two well-known AutoML packages are *Auto-Sklearn* and *H2O AutoML*.

2.5 Object Tracking

Object tracking is an application of deep learning to continuously estimate the position and orientation of an object in a video sequence. There are two types of video tracking: *Single Object Tracking* (SOT) and *Multiple Object Tracking* (MOT). SOT focuses only on one unique target object and keeps track of that throughout a video sequence. MOT tracks multiple objects, generates unique IDs for each detected object, and tracks their position as they move from one frame to the next. The system can maintain the ID assignment for each object, allowing it to keep track of the number of objects present at any given time. With tracking information, states such as position, velocity, and acceleration can be estimated [7].

2.5.1 SORT Algorithm

The Simple Online and Real-time Tracking (SORT) algorithm is a type of MOT designed to efficiently track multiple objects in 2D images for online and real-time applications. SORT is made of 4 key components: *detection*, *estimation*, *data association*, and *creation and deletion of track identities*.

Detection

Detection is the first step in the SORT tracking module. An object detector, like Stereo R-CNN, detects objects in the frame that are intended for tracking. The detections are then passed on to the next step, estimation.

Estimation

The estimation model propagates the detections from the current image frame to the next, estimating the state of the target in the next frame. Equation 2.12 presents the state of each target:

$$\mathbf{x_k} = \begin{bmatrix} u & v & s & r & \dot{u} & \dot{v} & \dot{s} \end{bmatrix}, \tag{2.12}$$

where *u* and *v* represent the pixel location of the center of the target in x- and y-direction, while *s* and *r* represent the scale and the aspect ratio of the target's bounding box [4].

The *Kalman filter* is an example of a commonly used estimation model. It works in two steps: the *prediction step* and the *update step*. The prediction step utilizes a model to estimate the object's position at the next image frame based on its previous trajectory. One commonly used model for this purpose is the *constant velocity model*. The constant velocity model is a physical model that assumes an object moves at a constant speed in a straight line. The update step is initiated when an object is detected and associated with a target. The Kalman filter uses the detected bounding box to update the estimate of the object's state. If a detection is not associated with the target, the linear velocity model predicts its state without adjustments.

Data Association

After the estimation step, an assignment cost matrix is computed to establish associations between detections and existing targets. This cost matrix is based on the Intersection-over-Union (IoU) distances, which measure the overlap between each detection and all predicted bounding boxes from the existing targets. The assignment is rejected if the IoU of detection and target is less than an imposed threshold, IoU_{min}. This rejection step ensures that only accurate and reliable associations are made between detections and targets.

Creation and Deletion of Track Identities

This module handles creating, deleting, and maintaining IDs for object tracking in the scene. Unique IDs are generated for objects and are continuously tracked based on the IoU_{min} threshold. If a detection has an overlap with an existing target equal to or greater than IoU_{min}, it retains its assigned ID. If a detection overlaps less than IoU_{min}, a new ID is assigned to the detection.

Tracks are terminated if not detected for a specified number of frames, known as the T_{lost} threshold. If the object reappears in the scene after being lost, tracking resumes implicitly under a new ID [10].

3

Materials and Methods

This chapter presents the materials and methodologies employed throughout the master project. Specifically, it encompasses the development and implementation of a system designed to detect H_2S concentrations based on video of salmon juveniles. Figure 3.1 presents a simplified representation of the system from a user's perspective. It showcases the integration of video input, data processing, and the resulting H_2S concentration as output. This visual depiction offers an overview of how the system operates and highlights the key steps involved in detecting and obtaining H_2S concentrations.

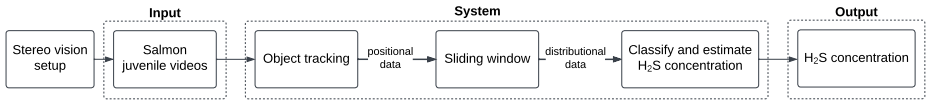


Figure 3.1: The H_2S detection system from the user's point of view, with three modules: input, system, and output.

The utilized materials and methods are partially based on previous work. Therefore, some sections concerning the camera setup, materials, and methodologies utilized were also mentioned in the Specialization Project [14]. This concerns Section 3.1, 3.2, 3.3, and Subsection 3.4.1, 3.5.1, and 3.5.2.

3.1 Stereo Vision Setup

A pair of Alvium cameras were mounted as a parallel stereo setup in a RAS tank containing salmon juveniles during SINTEF Ocean's H_2S experiment. The camera specifications are presented in Table 3.1. Multiple videos were captured to gather data on fish behavior throughout the experiment.

Table 3.1: Camera specifications of the cameras used for the stereo vision setup.

Camera specifications	
Sensor type	Alvium 1500 C-510 NIR
Resolution	2592×1944 $(width \times height)$
Sensor type	CMOS
Sensor size	Type 1/2.5
Pixel size	$2.2\mu m \times 2.2\mu m$
Max. frame rate	68 fps
Lens	C Series VIS-NIR
Focal length	3.50mm
Field of view	Horizontal: 41.2mm - 102.4° Vertical: 26.8mm - 82.3° Diagonal: 63.6mm - 117°

3.2 Hardware and Software

This section presents the hardware and software used in this project.

3.2.1 CUDA Toolkit

CUDA Toolkit is a parallel computing platform and programming model developed by NVIDIA for computing on its own graphics processing units (GPU). CUDA enables computer-intensive applications to speed up by taking advantage of the parallel processing power of GPUs [11]. The training of Stereo R-CNN and object tracking utilized this tool.

GPU Access

Specific components of the software required NVIDIA GPU access due to their usage of CUDA. A computer with the required hardware power was provided by NTNU and accessed via *Secure Shell* (ssh). The computer contained 3 NVIDIA GeForce RTX 3090 GPUs with CUDA Toolkit 11.8.

3.2.2 GitHub

GitHub is a web-based platform used for software development. It allows multiple people to work on projects simultaneously, making it easier for teams to collaborate and keep track of changes. The master project repository can be found on GitHub[1].

3.2.3 Tools and Libraries

Anaconda was utilized to create an environment to install the required libraries. Anaconda provides the opportunity to create environments of different versions of Python and install, remove, and upgrade packages within them. The environment used for this project was Python 3.6.13. A list of essential libraries and their used versions is displayed below.

- *PyTorch 1.10.1*
- *Imgaug 0.4.0*
- *OpenCV 4.6.0*
- *Scikit-Learn 1.0.2*
- *H2O 3.40.0.2*

[1] Repository of the project: https://github.com/TTK4900

3.3 Data Provided by SINTEF Ocean

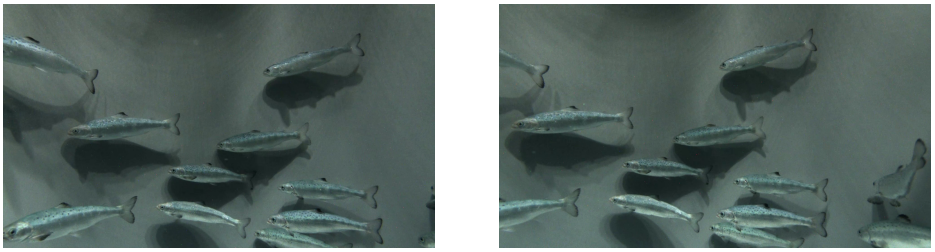
SINTEF Ocean provided datasets in the form of videos. Several videos of salmon juveniles were provided. Two videos of a chessboard were also given for calibration purposes. All videos were captured in the RAS tank with the same stereo vision setup. The image frame was of pixel dimension 1920×2160 ($width \times height$), containing images captured by the left and right camera. The pixel dimension of one camera was 1920×1080.

3.3.1 Video Data of Salmon Juveniles

The provided salmon juvenile videos were captured during SINTEF Ocean's H_2S experiment. The video data contained footage over 11 days where the RAS tank was exposed to different concentrations of H_2S. Each day's dataset contained between 9-43 videos, each with an approximate duration of 15 minutes. Table 3.2 presents an overview of the provided videos from each day, the start of recording, the start of H_2S dosing, and the duration. Figure 3.2 shows a stereo image sample from a salmon juvenile video.

Table 3.2: Overview of provided video data of salmon juveniles from the H_2S experiment.

Date	**Start of recording**	**Start of H_2S dosing**	**Approximate recording length** (hours:min)	**Amount of videos per day**
27.06.2022	15:30	-	16:00	64 videos
28.06.2022	07:18	08:25	4:15	17 videos
29.06.2022	07:00	08:03	4:00	16 videos
30.06.2022	07:00	08:03	3:00	12 videos
01.07.2022	06:45	08:00	3:00	12 videos
02.07.2022	06:40	08:05	3:15	13 videos
03.07.2022	06:45	07:52	3:30	14 videos
04.07.2022	06:45	08:00	2:30	10 videos
05.07.2022	06:45	07:50	2:15	9 videos
06.07.2022	06:48	07:50	3:00	12 videos
07.07.2022	06:42	07:50	5:15	21 videos

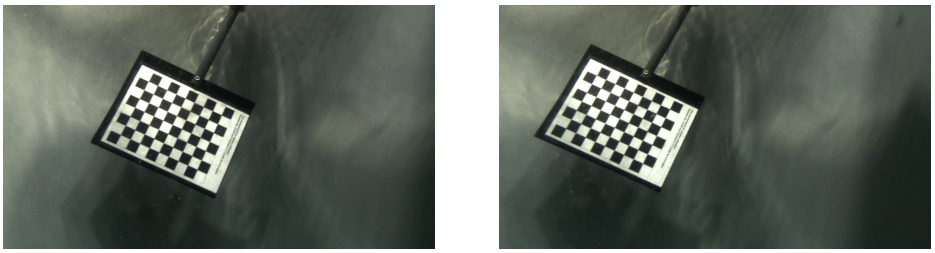


(a) Snapshot from left camera. **(b)** Snapshot from right camera.

Figure 3.2: Snapshot of a salmon juvenile video from the dataset.

3.3.2 Video Data of Chessboard

A dataset containing videos of a chessboard was provided for camera calibration. Figure 3.3 shows a stereo image sample of one of the videos of the chessboard.



(a) Snapshot from left camera. **(b)** Snapshot from right camera.

Figure 3.3: Snapshot of a chessboard video from the dataset.

Stereo Camera Calibration

Stereo camera calibration was performed during the Specialization Project to extract 3D information from stereo images containing salmon juveniles. The method employed was Zhang's approach [37], which utilizes images of a chessboard pattern to determine both the intrinsic and extrinsic parameters of the cameras involved. It was unnecessary to perform the calibration anew, given the use of identical video data in both the Master's Thesis and the Specialization Project. Consequently, the calibration results from the Specialization Project was adopted. For an in-depth comprehension of the underlying theory, methodological approach, and corresponding results, see Section 2.3.2, 3.6, and 4.2 in Project Thesis [14].

3.3.3 Measurements from the H_2S Experiment

During the experiment, an instrument simultaneously measured salinity, temperature, oxygen, carbon dioxide, and H_2S. These measurements were provided in the form of a `.csv` file. Table 3.3 presents the highest measured concentration of H_2S for each day and the corresponding time.

Table 3.3: H_2S measurements during the H_2S experiment.

H_2S measurements		
Date	**Time of max. concentration**	**Max. measured H_2S concentration**
27.06.2022	-	-
28.06.2022	08:41	$3.54\mu g/L$
29.06.2022	08:18	$8.13\mu g/L$
30.06.2022	08:22	$9.72\mu g/L$
01.07.2022	08:45	$5.96\mu g/L$
02.07.2022	08:27	$3.55\mu g/L$
03.07.2022	08:18	$7.27\mu g/L$
04.07.2022	08:30	$14.45\mu g/L$
05.07.2022	08:16	$32.15\mu g/L$
06.07.2022	08:17	$67.72\mu g/L$
07.07.2022	08:10	$64.43\mu g/L$

3.4 Preparing Image Data

A dataset of 380 stereo images was created from 10 of the salmon juvenile videos during the Specialization Project. One of the main limitations of the annotating process was the time-consuming job of manually annotating all the fish within an image. Moreover, the annotation process would not indicate how well the trained Stereo R-CNN model did or what types of images were lacking from the dataset. Therefore, it was of interest to implement a new annotation strategy that would accelerate the process in addition to getting feedback on model performance.

3.4.1 Correction of Disparity Between the Left and Right Image

The Stereo R-CNN required that objects in the corresponding left and right image pair were vertically aligned. The videos captured by the stereo camera setup had some disparity between the left and right image frames. The disparity was found to be 25 pixels in y-direction. The left image was cropped 25 pixels in y-direction at the top, and the right image was cropped 25 pixels at the bottom. Figure 3.4 visualizes the alignment. All images utilized for training the Stereo R-CNN and images fed into the model for object detection were resized. Consequently, the images attained a new resolution of 1920×1055 pixels.

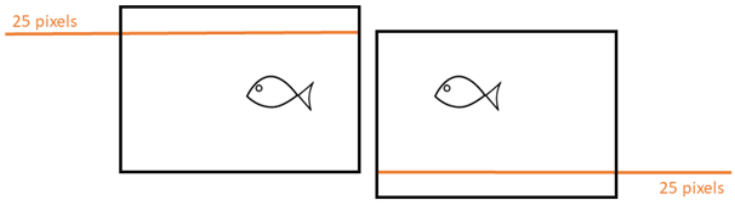


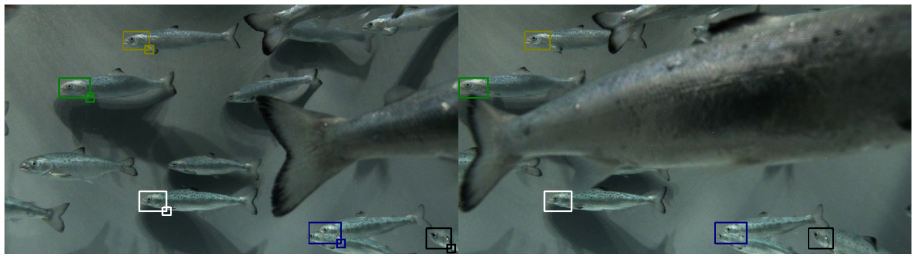
Figure 3.4: Correction of disparity in y-direction between left and right image.

3.4.2 Integrating Stereo R-CNN in the Annotation Strategy

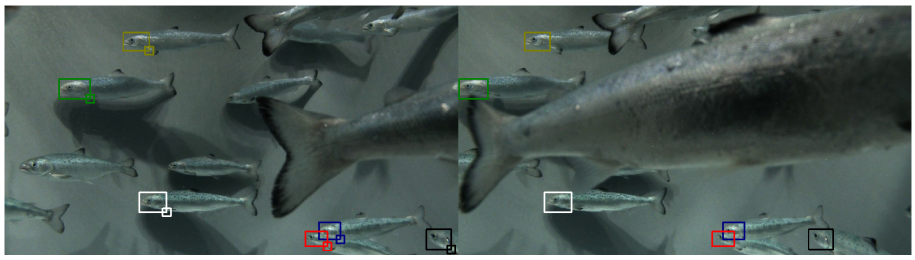
The new annotation strategy involved using a pretrained Stereo R-CNN model to detect bounding boxes around salmon juveniles within an image. The images selected for annotation were subsequently displayed with bounding boxes encompassing all detected fish by the model. The annotation process was streamlined by reducing the manual work to oversee and fine-tune the images. Figure 3.5 illustrates an example of the new annotation process.

Implementation

Due to limitations of GPU access, it was of interest to implement the annotation tool to utilize CPU. The Stereo R-CNN model had a slow inference speed on CPU. It would take approximately 1 minute to process one image to detect the fish within it. The processed image would then be displayed for annotation, and only after manual annotation would the Stereo R-CNN be fed the next image. This sequential process proved to be relatively inefficient. To solve this, a *multiprocessing* strategy was introduced. Multiprocessing is a Python package that supports the spawning of processes, effectively allowing for simultaneous operations.



(a) Detected objects of Stereo R-CNN.



(b) Corrected objects.

Figure 3.5: Example of the annotation process with integrated Stereo R-CNN.

Multiprocessing with a queue was implemented. The model would then process images continuously in the background, storing the detected bounding boxes of an image in a queue. This way, the manual annotation could be performed concurrently and thereby streamlining the overall process. The annotation tool was developed to iterate through all of the provided salmon juvenile videos to include all types of scenarios and lighting conditions.

3.4.3 Annotation Process

The creation of the stereo image dataset was an iterative process. Including a pretrained Stereo R-CNN model in the annotation tool facilitated an assessment of the model's performance in fish detection. This generated an iterative feedback loop, wherein images with wrongly detected fish were adjusted and added to the dataset, and correctly detected images were dismissed. After adding some images to the dataset, the Stereo R-CNN model was retrained, and training and validation loss were assessed over multiple epochs to identify the best model. The updated model was set in the annotation tool. The process was repeated until the validation loss implied that the model had reached its performance limit. Figure 3.6 presents the workflow of the annotation process and training of Stereo R-CNN.

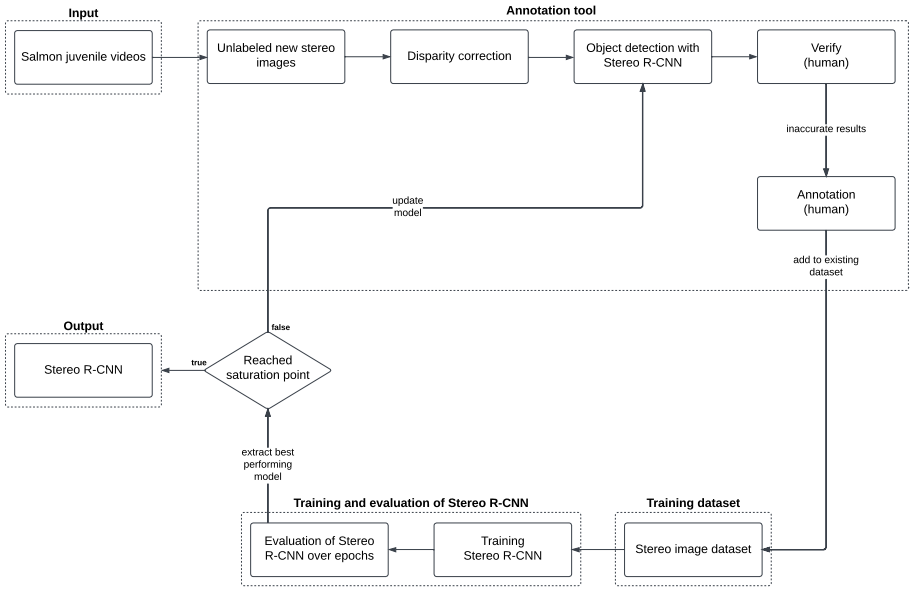


Figure 3.6: A flow diagram of the procedure for the annotation process and training of Stereo R-CNN. The procedure is split into five modules: input, annotation tool, training dataset, training and evaluation of Stereo R-CNN, and output. This process was repeated until the end results were satisfactory.

3.4.4 Image Augmentation

Observations during the annotation process revealed a lack of diversity in the dataset. The augmentation methods *multiply*, *linear contrast*, *rotation*, *horizontal flip*, and *Gaussian blur* were applied during the Specialization Project. It was discovered that rotation and horizontal flip were sources of inaccurately training the model. Rotation could lead to rotating the annotated fish out of the image, and horizontal flip led to wrong locations of the fish in the left and right image relative to each other. Due to this, only multiply, linear contrast, and Gaussian blur were used. Images of the different augmentation methods are presented in Figure 3.7, 3.8, and 3.9, respectively. The leftmost image in the figures is the original image, followed by augmented images. The green boxes present the annotated regions.

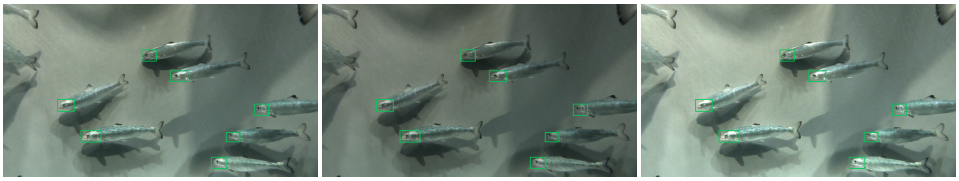


Figure 3.7: Augmentation method: *Multiply*.

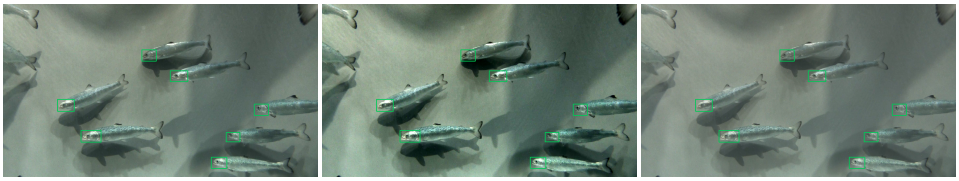


Figure 3.8: Augmentation method: *Linear contrast*.

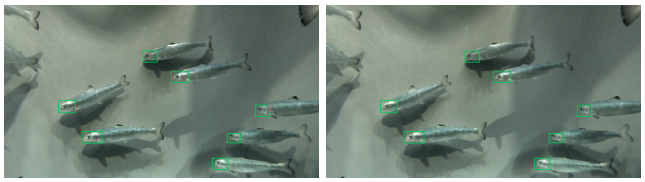


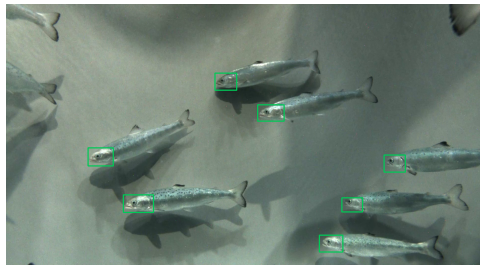
Figure 3.9: Augmentation method: *Gaussian blur*.

Combination of Data Augmentation Methods

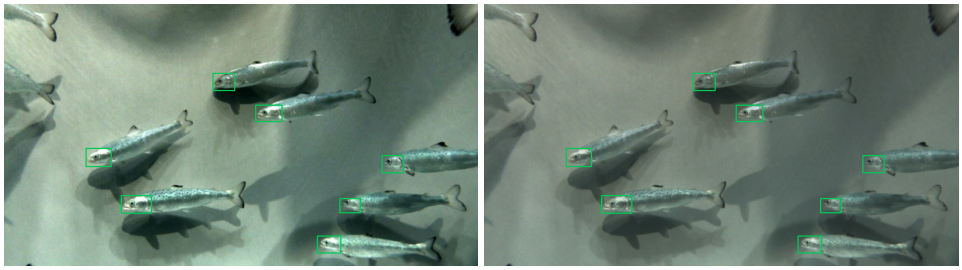
A combination of the augmentation methods presented above was used to create images from each sample in the training dataset. The specific combinations are presented in Table 3.4. Figure 3.10 provides two augmented images derived from a single image sample.

Table 3.4: Applied combination of data augmentation methods.

Augmentation method	Description, occurrence, and values
Multiply	Make the image darker or lighter. Multiply all pixels in an image by a random value between 0.9 and 1.1.
Linear contrast	Increasing or decreasing contrast in the image. Each pixel in the image is scaled with a value between 0.8 and 1.4.
Gaussian blur	Blurring of images. Applied to 70% of all image samples, with a value between 0 and 1.3.



(a) Original image.



(b) Image after applying a combination of data augmentation methods.

Figure 3.10: Combination of data augmentation methods.

3.4.5 Image Dataset for Stereo R-CNN

In addition to the 380 stereo images annotated during the Specialization Project, 1062 new images were annotated with the new annotation strategy. The resulting dataset consisted of 1442 stereo images and a `.csv` file containing coordinates of corresponding bounding boxes. The number of salmon juveniles in an image

ranged between 1-10.

A separate test set was created to evaluate the Stereo R-CNN model's actual performance. This dataset was created without using the new annotation tool to avoid biased performance measures of testing different models. It comprised 66 images with a corresponding `.csv` file.

3.5 Training of Stereo R-CNN

The dataset presented in Subsection 3.4.5 was employed to train and assess the Stereo R-CNN. The split ratio of the training dataset was 80:20, meaning that 80% went to the training of the model and 20% went to validate the model. The samples were distributed randomly.

The Stereo R-CNN was trained with and without data augmentation. In the case of data augmentation inclusion, the augmentation combinations were only applied to the training dataset. Three augmented images were created of each sample in the training set and added to the dataset. The validation and test dataset contained only original annotated images, keeping it as accurate to the true data as possible.

3.5.1 Loss Function of Stereo R-CNN

The loss function used in the training process of Stereo R-CNN is presented in Equation 3.1. L^p and L^r represents RPN and R-CNN loss respectively. Bounding box regression is referred to as *reg*, classification as *cls*, and loss of 2D boxes as *box*. The loss function is a combination of *Logarithmic loss* and *Smooth L1 loss*. Logarithmic loss is based on the probability of correctly classifying the regions, and Smooth L1 loss represents how well the predicted bounding boxes match the true bounding boxes [31].

$$L = L^p_{cls} + L^p_{reg} + L^r_{cls} + L^r_{box} \tag{3.1}$$

3.5.2 Training Steps of Stereo R-CNN

A computer with NVIDIA hardware accessed via *ssh* was utilized during the training of the Stereo R-CNN model. The steps taken towards training the Stereo R-

CNN model were as follows:

1. An anaconda environment was created and activated to install the required libraries:
   1.1 *PyTorch 1.10.1* and *torchvision 0.11.2* was installed with CUDA 11.3.
   1.2 Other required libraries listed in `requirements.txt` were installed.
2. The model environment was built by running the command line: `python setup.py build develop`
3. A pretrained ResNet-101 model was downloaded and placed in the directory: `data/pretrained_model`
4. The created dataset consisting of stereo image pairs and the corresponding `.csv` file was loaded in the directory: `data/training_data`
5. All project files were transferred to the NVIDIA computer via an *ssh* connection.
6. Hyperparameters of the model were set in the `trainval_net.py` file. This includes *learning rate*, *number of epochs*, and *batch size*.
7. The training of Stereo R-CNN was conducted by running the command line: `python trainval_net.py`

The Stereo R-CNN model was saved at every 10 epoch interval during training. By saving the model during training, it was possible to test the model at different epochs and easily retrieve the model with the best performance. The training process could also be started again at a given epoch in case of process disruption.

3.5.3 Evaluation of Stereo R-CNN Performance

Following the training of the Stereo R-CNN model, an evaluation was conducted on models from various epochs. The models were tested utilizing the test dataset. Three evaluation metrics were employed to assess model performance: *Median IoU*, *Precision*, and *Recall*. IoU, as defined in Equation 3.2, represents the degree of overlap between the true and the predicted bounding box. An IoU threshold was established to classify the predicted boxes as either *True Positives* (TP) or *False Positives* (FP). This value was set to 0.4, indicating that boxes with at least 40% overlap would be considered TP, while those with less than 40% overlap

would be labeled FP. Precision quantified the accuracy of the model's predictions, specifically, the proportion of the predicted bounding boxes correctly identified as fish. Recall measured the percentage of all true bounding boxes the model successfully detected. Precision and recall are defined in Equation 3.3 and 3.4, respectively.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \tag{3.3}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \tag{3.4}$$

3.6 3D Coordinate Estimations of Detected Objects

3D coordinates of detected objects could be estimated after training the Stereo R-CNN. Utilizing the camera calibration outcomes from the Specialization Project, it was of interest to investigate whether an improved Stereo R-CNN model influenced the estimation of 3D coordinates for detected objects. The same 10 images employed during the testing phase of the Specialization Project were utilized to compare the differences.

3.7 Object Tracking

Object tracking of video material could be performed after establishing the best-performing Stereo R-CNN model and testing of 3D coordinate estimation. Figure 3.11 presents an overview of the object tracking process flow, from salmon juvenile video as input to positional data output.

Object tracking was initially implemented to track fish over a specified number of image frames in a video. It was modified to accept a sequence of videos as input and iterate through each image frame in every video, providing 3D coordinates for each detected object. The SORT algorithm assigned every detected object a unique ID to track it from one image frame to the next. The estimation step in the SORT algorithm utilized a Kalman filter with a constant velocity model as its prediction

step. The IoU_{min} threshold was set to 0.3, meaning that a detected object and the predicted bounding box of a target had to overlap at least 30% in order to maintain its ID.

The tracking module gathered 3D coordinates for a number of frames, namely *frame count* (f_c), before feeding them into a Kalman filter. Based on these coordinates, the filter estimated a trajectory comprising the 3D position and its decomposed velocity. The filter output provided fish position (x, y, z) and velocity (v_x, v_y, v_z). Furthermore, the velocity magnitude, or speed, was calculated according to Equation 3.5. The salmon juvenile videos were captured at 15 frames per second. By setting the frame count to 15, the Kalman filter would estimate positional data every second.

The rate of speed change was determined based on estimated velocities over time. If the frame count was set to 15, a fish needed to be tracked for at least 2 seconds to calculate its speed change rate, given that the Kalman filter estimated velocity every second. The calculation of the speed change rate is given in Equation 3.6, where Δv represents the change in speed and Δt denotes the change in time. The term fps represents the frames per second rate of the utilized video data, which was set to 15. With a frame count of 15 frames, the value of $\Delta t = 1$.

The obtained 3D position (x, y, z), decomposed velocity (v_x, v_y, v_z), velocity magnitude ($|v|$), and speed change rate (r_s) were acquired from object tracking and subsequently stored in `.pickle`-files.

$$|v| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{3.5}$$

$$r_s = \frac{\Delta v}{\Delta t} = \frac{|v_2| - |v_1|}{frame\ \ count/fps} = \frac{|v_2| - |v_1|}{frame\ \ count/15} \tag{3.6}$$

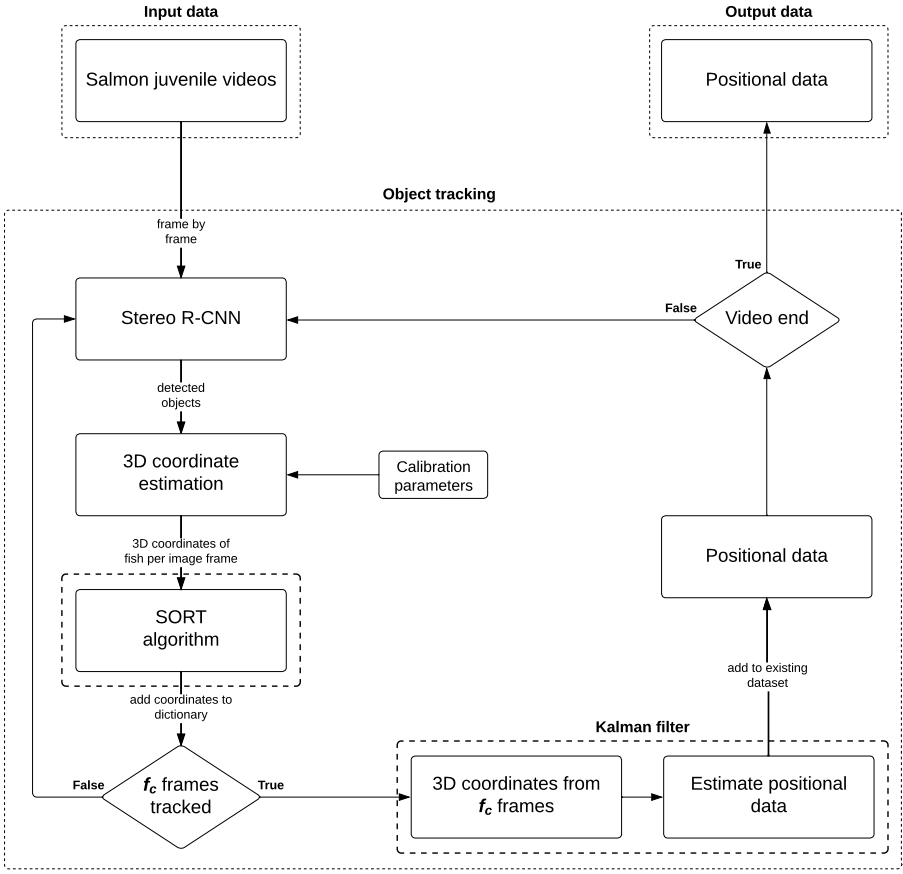


Figure 3.11: A flow diagram of the object tracking process, with four main modules: input data, SORT algorithm, Kalman filter, and output data.

3.7.1 Video Data Utilized for Object Tracking

Throughout the Specialization Project, 16 videos were employed to gather positional data from two behavioral states of the fish, namely *normal behavior* and *stressed behavior*. There was interest in examining more video material to construct a more extensive and comprehensive dataset. Consequently, all provided videos from SINTEF Ocean were utilized to acquire positional data for further analysis.

3.7.2 Obtaining Extended Time Series of Positional Data

A primary limitation during the Specialization Project was the short time series of fish position, velocity, and speed change rate. With the initial threshold value of $T_{lost} = 1$, the SORT algorithm assigned salmon juveniles new IDs upon being lost for more than one image frame. To achieve longer time series for fish, this value was experimentally adjusted to 5, 10, and 20, allowing the fish to be lost for a greater number of frames before being considered lost.

Additionally, various frame counts, f_c, were examined with $T_{lost} = 5$. The frame count was assessed at 5, 15, and 30. The different values signified that the Kalman filter estimated position every 1/3 of a second for $f_c = 5$, every second for $f_c = 15$, and every 2 seconds for $f_c = 30$. Table 3.5 presents the different combinations of T_{lost} and frame count that were tested.

It was determined by visual inspection that the most satisfactory results were obtained with $T_{lost} = 5$ and $f_c = 15$. Consequently, these values were selected for further analysis.

Table 3.5: Overview of combinations of tested values for T_{lost} threshold and frame count.

Threshold	Frame count
$T_{lost} = 1$	15 frames
$T_{lost} = 5$	5 frames
$T_{lost} = 5$	15 frames
$T_{lost} = 5$	30 frames
$T_{lost} = 10$	15 frames
$T_{lost} = 20$	15 frames

3.8 Processing Acquired Positional Data

Processing acquired positional data from object tracking was utilized to examine potential features relevant to the task of detecting H_2S. The processing encompassed visualization of the positional data and the creation of various distribution datasets to test and assess which is best suited for the task.

3.8.1 Visualizing Positional Data

Visualizing the positional data was a step towards examining fish behavior and identifying the most suitable features for H_2S detection. The positional data was plotted against the corresponding measured H_2S concentration over the same time period. Figure 3.12 illustrates the measured H_2S concentration, and Figure 3.13 presents positional data from *07.07.2022*.

The positional graphs represent the mean value of all the data points collected within a one-minute interval. The shaded area around the graphs indicate the variation within the same minute. The shaded region includes 80% of the data points, excluding the lower and upper 10% of the data points. The red area marks the time frame with high H_2S concentration and where the positional data appeared to deviate. For plots of *27.06.2022-06.07.2022*, see Appendix A.

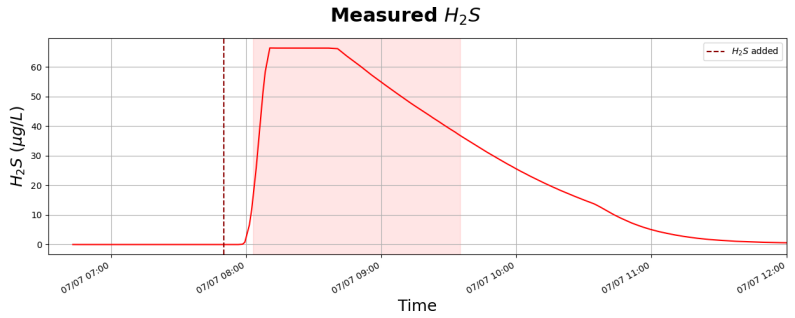


Figure 3.12: *07.07.2022*: Measured H_2S.

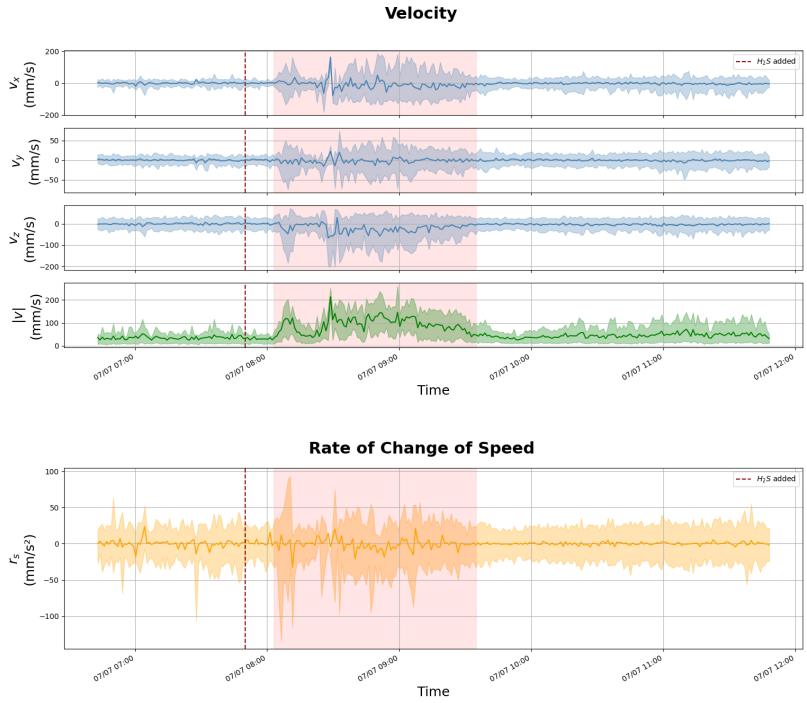


Figure 3.13: *07.07.2022*: Velocity and speed change rate data acquired from object tracking.

3.8.2 Creating Distribution Datasets

Datasets of positional data were created to classify and estimate H_2S concentrations. In order to obtain a representative sample of the fish population, it was decided to use normalized positional data distributions as samples in the dataset. It was decided to include decomposed velocity, velocity magnitude, and speed change rate distributions to provide as much information related to fish behavior as possible.

Sliding Window

A sliding window technique was utilized to create the positional data distributions. All data points within a specified *window size* were selected, and the values were distributed. The window was then moved sequentially to capture the next set of data points, and the distributions were recalculated. The *step size* determined the

window's movement through the dataset. A smaller step size resulted in a higher degree of overlap between consecutive windows, and a larger step size led to less overlap. An illustration of the technique is presented in Figure 3.14.

The start and end time was set manually by examining the measured H_2S concentrations and positional data. These times correspond to the start and end of the shaded red area in the positional plots. As the sliding window moved, distributions of positional data were added to the distribution dataset. The complexity of the distributions was influenced by the number of *bins*, b_n, chosen for histogram representation. A higher number of bins gave a more detailed distribution, revealing subtler patterns and nuances within the data. Conversely, fewer bins simplified the distribution, potentially obscuring some finer details but providing a more generalized presentation.

One data point in the dataset consisted of distributions of decomposed velocity, velocity magnitude, and speed change rate. Two different data samples are illustrated in Figure 3.15 and 3.16, presenting positional data from *27.06.2022* (low activity) and *07.07.2022* (high activity), respectively. The histograms were made with a window size of 10 minutes.

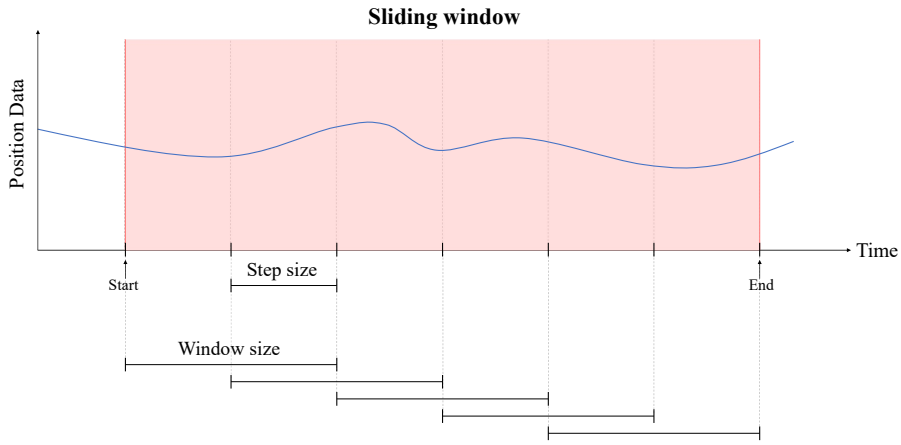


Figure 3.14: Sliding window technique for collecting distribution data.

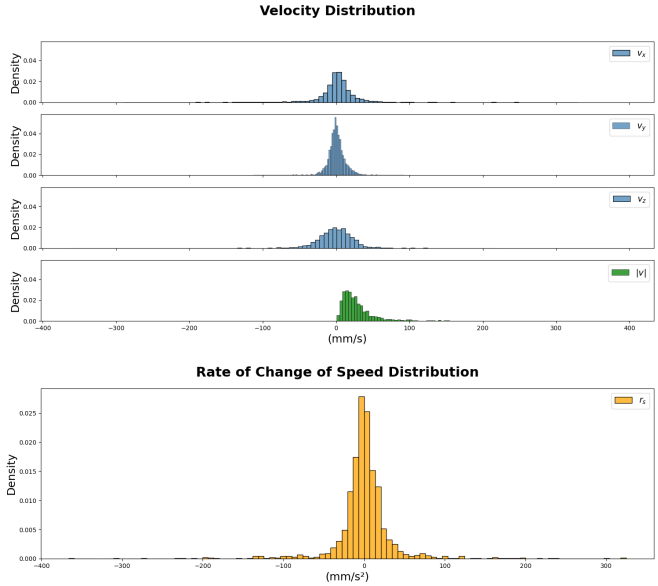


Figure 3.15: Example of a data sample consisting of velocity and speed change rate distributions from *27.06.2022* labeled with $0\mu g/L$. $b_n = 100$.

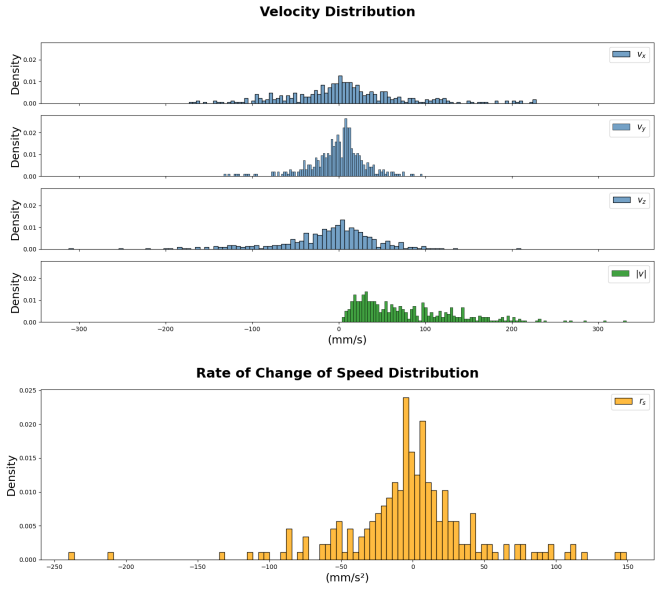


Figure 3.16: Example of a data sample consisting of velocity and speed change rate distributions from *07.07.2022* labeled with $66.4\mu g/L$. $b_n = 100$.

Distribution Datasets

Five combinations of window size (w_s), step size (s_s), and number of bins (b_n) were set to create five distribution datasets. These combinations are presented in Table 3.6.

Distributions based on positional data from each day of the H_2S experiment were collected. The number of created data samples for each day varied based on the sliding window's time frame, window size, and step size. The distributions were labeled with corresponding measured H_2S concentration. Table 3.7 presents the time frame used for each day, created distribution samples, and their labels.

Two window sizes of 5 and 10 minutes were tested. The quantity of positional data points utilized to define the distributions depended on the selected window size and the specific day of the H_2S experiment. Table 3.8 presents the average number of velocity samples, N_{avg}, used to define a distribution for each day, dependent on window size. It also shows the minimum, N_{min}, and maximum number of samples, N_{max}, used to define distributions.

Some samples were selected from the datasets to create test sets. The test sets contain at least one sample from each day, representing all H_2S concentration labels. Overlapping samples in the training dataset were deleted, ensuring that positional data used to create a distribution would not be used in both the training and test set. Table 3.6 presents the length of the resulting training and test datasets.

Table 3.6: Created datasets with the sliding window technique.

Overview of distribution datasets					
Dataset	**Window size**	**Step size**	**Number of bins**	**Training size**	**Test size**
1	5 min	2.5 min	50	237 samples	16 samples
2	5 min	2.5 min	100	237 samples	16 samples
3	10 min	2.5 min	50	208 samples	13 samples
4	10 min	5 min	50	113 samples	13 samples
5	10 min	5 min	100	113 samples	13 samples

Table 3.7: Overview of time frame used to collect distribution samples and corresponding created distribution data samples for each day, including their labels.

Date	**Start and end time for collecting data**	**Time frame** (hours:min)	**Created data samples**	**Label**
			Window size = 5/10 min ($s_s = 2.5/5$ min)	
27.06.2022	16:28-19:08	02:40	87 / 41	$0\mu g/L$
28.06.2022	08:33-08:03	00:30	6 / 3	$3.5\mu g/L$
29.06.2022	08:09-08:39	00:30	6 / 3	$8\mu g/L$
30.06.2022	08:15-08:45	00:30	6 / 3	$9.5\mu g/L$
01.07.2022	08:09-09:09	01:00	19 / 8	$5.8\mu g/L$
02.07.2022	08:14-09:09	00:55	16 / 7	$3.5\mu g/L$
03.07.2022	08:00-09:10	01:10	19 / 10	$7.2\mu g/L$
04.07.2022	08:10-09:10	00:50	18 / 8	$14.4\mu g/L$
05.07.2022	08:03-09:03	01:00	16 / 9	$32\mu g/L$
06.07.2022	08:06-09:01	00:50	14 / 8	$67.7\mu g/L$
07.07.2022	08:03-09:43	01:40	31 / 13	$66.4\mu g/L$

Table 3.8: Overview of the minimum, maximum, and average number of velocity samples used to define velocity distributions.

Date	Velocity samples					
	Window size = 5 min			Window size = 10 min		
	N_{min}	N_{max}	N_{avg}	N_{min}	N_{max}	N_{avg}
27.06.2022	487	1662	1197	1243	2968	2404
28.06.2022	578	1425	907	1190	2405	1806
29.06.2022	1106	1665	1523	2610	3285	3078
30.06.2022	1258	1966	1471	2549	3387	2936
01.07.2022	463	902	690	1079	1728	1392
02.07.2022	141	1431	538	430	1776	1092
03.07.2022	571	1004	819	1211	1872	1621
04.07.2022	370	3092	2227	370	6098	4146
05.07.2022	476	2091	1389	1055	3873	2741
06.07.2022	145	1055	666	385	2271	1426
07.07.2022	25	695	286	75	1503	616

3.9 H_2S Detection using Machine Learning

Several ML methods were assessed for detecting H_2S concentration based on distribution data. Six classification models and five regression models were implemented for the task. The models were trained and tested on the created datasets. The process is depicted in Figure 3.17, showcasing the stages of training and evaluating these models. The objective was to identify the most suitable model and the respective dataset that could accurately estimate H_2S concentration.

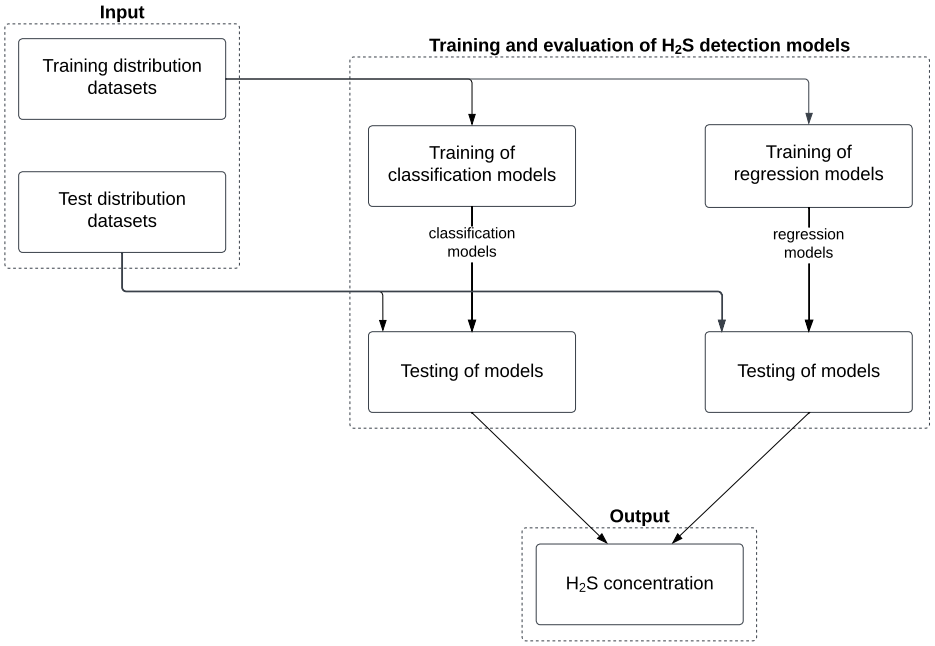


Figure 3.17: A flow diagram of the procedure for training and testing the classification and estimation models.

3.9.1 Classification of H_2S Concentration

Classification was the first method assessed for detecting H_2S concentration based on positional distribution data.

Model Candidates

Three SVCs with linear, RBF, and polynomial kernel were implemented utilizing the built-in function `SVC()` from the *scikit-learn* library. All kernels were trained with the default regularization parameter $C = 1$. The default gamma parameter of the RBF and polynomial kernel was also used:

$$\gamma = scale = \frac{1}{n_{features} \times X.var()}, \tag{3.7}$$

where $n_{features}$ is the number of features and $X.var()$ is the variance of the

dataset. The degree of the polynomial kernel was set to $d = 3$.

A decision tree and a random forest classifier were also implemented and tested with default values from the same library. The decision tree algorithm utilized by the *scikit-learn* library is CART. The random forest had the value of $n\_estimators = 100$, meaning there are 100 trees in the forest. Other default values applied by both models were as follows:

- $max\ depth =$ *None*: The nodes will expand until all leaves are pure or until all leaves contain less than $min\_samples\_split$ samples.
- $min\_samples\_split = 2$: The minimum number of samples required to split an internal node was 2.
- $min\_samples\_leaf = 1$: The minimum number of samples required at a leaf node was 1.

An automatic machine learning tool for classification, *Auto-Sklearn*, was utilized to strive for optimal classification results. This method automatically searches for the best model and hyperparameters for a given classification problem. Auto-Sklearn offered some interpretability and transparency by providing detailed logs and output. These logs describe the selected models, their hyperparameters, performance metrics, and evaluation criteria used for model comparison.

A summary of the implemented models and their documentation for the classification of H_2S concentration:

- `SVC(kernel='linear')`[2]
- `SVC(kernel='rbf')`[2]
- `SVC(kernel='poly', degree=3)`[2]
- `DecisionTreeClassifier()`[3]
- `RandomForestClassifier()`[4]
- `AutoSklearnClassifier()`[5]

[2] scikit-learn.org/SVC/
[3] scikit-learn.org/DecisionTreeClassifier/
[4] scikit-learn.org/RandomForestClassifier/
[5] automl.github.io/auto-sklearn/classifier/

Evaluation of Classification Performance

The classification performance of the different models was measured in *accuracy*, defined in Equation 3.8. In addition, a confusion matrix was displayed to visualize the classification results of the test set.

$$\begin{aligned} accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ &= \frac{Number\ of\ correct\ estimations}{Total\ number\ of\ estimations} \end{aligned} \tag{3.8}$$

3.9.2 Estimation of H_2S Concentration

The second method assessed for detecting H_2S concentration was estimation. The aim was to examine if H_2S concentration could be estimated from positional data rather than classify it, considering H_2S concentration being a continuous variable.

Model Candidates

A SVR, a decision tree regressor, and a random forest regressor were employed utilizing the *scikit-learn* library. The SVR model was implemented with RBF kernel with default parameters $C = 1$ and $\gamma = scale$. The max_depth parameter of the random forest was set to 2. Other parameters of the random forest and the decision tree were implemented with the same default values as for classification.

Two automatic machine learning methods were also assessed for estimation with the aim of obtaining the best possible regression results. The automatic machine learning tool for regression from the *scikit-learn* library was implemented. Another automatic machine learning library called *H2O AutoML* was also employed. The difference between the two libraries lies in H2O AutoML's inclusion of deep learning networks as model candidates. Both methods provided interpretability and transparency of assessed models in terms of detailed logs of model leaderboards, hyperparameters, and performance metrics.

In summary, the following models were implemented for the estimation of H_2S concentration; each complemented with documentation:

- `SVR(kernel='rbf')`[6]

[6] scikit-learn.org/SVR/

- `DecisionTreeRegressor()`[7]
- `RandomForestRegressor(max_depth=2)`[8]
- `AutoSklearnRegressor()`[9]
- `H2OAutoML()`[10]

Evaluation of Estimation Performance

The estimation performance of the different models was measured in *R² score*, *mean absolute error* (MAE), and *root mean square error* (RMSE). R² score measures how well a regression model explains the variance in the target variable. It usually ranges from 0 to 1, with 1 indicating that the model perfectly explains the variance and 0 indicating that the model explains none. A higher R² score indicates a better model. Equation 3.9 presents the calculation of the R² score, where y represents the true value, $\hat{y}$ is the estimated value, and $\overline{y}$ is the average of all true values.

MAE is a measure of the average magnitude of errors between the estimated values and the true values. It is calculated by taking the average absolute differences between the estimated and true values. Lower values of MAE indicate a better model. Equation 3.10 defines the formula for calculating MAE.

RMSE presents the standard deviation of estimation errors and provides a measure of how well the model estimated the target value. RMSE is defined in Equation 3.11. It is sensitive to outliers as the differences are squared. It was easy to interpret as it had the same unit, $\mu g/L$, as the true and estimated values.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})}, \quad \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.9}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.10}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{3.11}$$

[7] scikit-learn.org/DecisionTreeRegressor/
[8] scikit-learn.org/RandomForestRegressor/
[9] automl.github.io/auto-sklearn/regressor/
[10] docs.h2o.ai/h2oAutoML/

Furthermore, a regression plot was displayed to represent the relationship between true and estimated values. Additionally to the regression line, a confidence interval was also visualized. The interval represents the range of values within. This range represents an uncertainty around the estimated values based on the variability of the data points around the regression line. This is the range of values one expects the estimations to fall between, with a confidence of 95%.

4

Results

This chapter presents results from the training and testing of Stereo R-CNN, the test of estimated 3D coordinates of detected objects, object tracking, and classification and estimation of H_2S concentration.

4.1 Training and Testing Stereo R-CNN

This section presents the training and testing results of Stereo R-CNN. The training process was done with and without data augmentation. The optimization algorithm *Adam* with *learning rate* 0.0001 was used to get the following results. The *batch size* was set to 1.

4.1.1 Training Without Data Augmentation

Figure 4.1 presents the training and validation loss from the final training process of Stereo R-CNN without data augmentation. It was trained over 100 epochs. The validation graph suggests that the minimum was achieved slightly before the 50^{th} epoch, after which the model started to overfit. The validation loss value at this point was approximately -21.

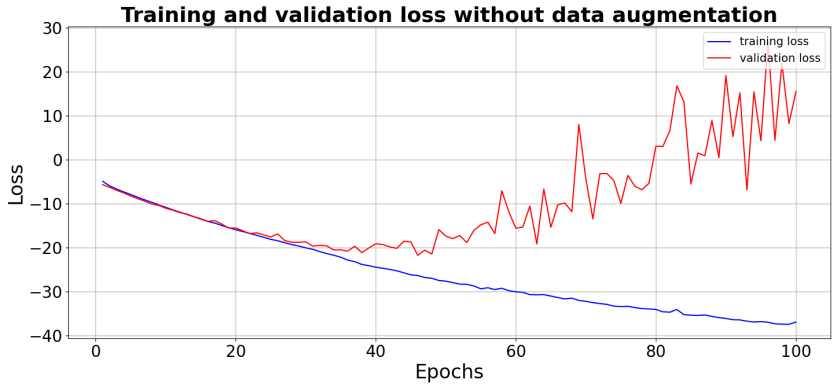


Figure 4.1: Training and validation loss without data augmentation.

4.1.2 Training With Data Augmentation

Figure 4.2 presents the training and validation loss of the training process over approximately 45 epochs of the Stereo R-CNN model with data augmentation. The plot indicates that the model was at its best at epoch 27, where the validation loss value was -20.

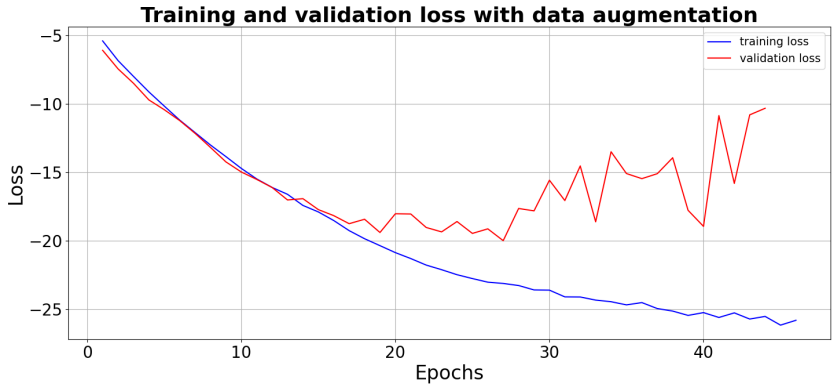


Figure 4.2: Training and validation loss with data augmentation.

4.1.3 Testing of Stereo R-CNN

The Stereo R-CNN models were tested utilizing the test set consisting of 66 images. Figure 4.3 and 4.4 show two example test stereo images. The green boxes are grounding truth boxes, and the red boxes are predictions. All fish in Figure 4.3 were detected and correctly classified. In Figure 4.4, every labeled fish was detected in addition to two FPs.

Model performance was measured with an IoU distribution of bounding boxes and the calculation of precision and recall for every stereo image in the test set. The Stereo R-CNN with and without data augmentation was tested at different epochs. Median IoU, precision, and recall results are presented in Table 4.1. The model used for further analysis was trained without data augmentation and extracted at epoch 50, marked with blue in the table. The corresponding IoU distribution is illustrated in Figure 4.5, with a median IoU of 87.4%. The average precision was 96.7%, and recall was 93.9%, presented in Figure 4.6.

Table 4.1: Overview of median IoU, precision, and recall obtained from testing Stereo R-CNN with and without data augmentation at different epochs.

Testing of Stereo R-CNN without data augmentation			
Model	**Median IoU**	**Precision**	**Recall**
Epoch 40	0.854	0.957	0.940
Epoch 50	0.874	0.962	0.939
Epoch 60	0.846	0.974	0.933
Epoch 80	0.890	0.979	0.910
Epoch 100	0.864	0.979	0.932

Testing of Stereo R-CNN with data augmentation			
Model	**Median IoU**	**Precision**	**Recall**
Epoch 10	0.820	0.915	0.898
Epoch 20	0.850	0.943	0.907
Epoch 30	0.848	0.965	0.913

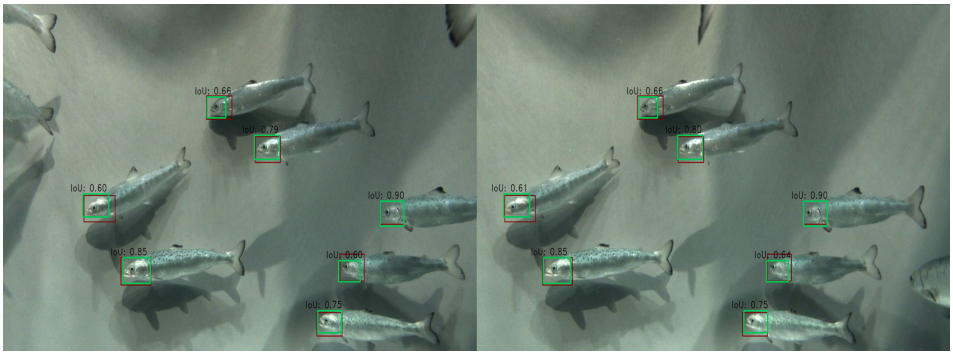


Figure 4.3: *Example test image 1*: Detections in a stereo image.

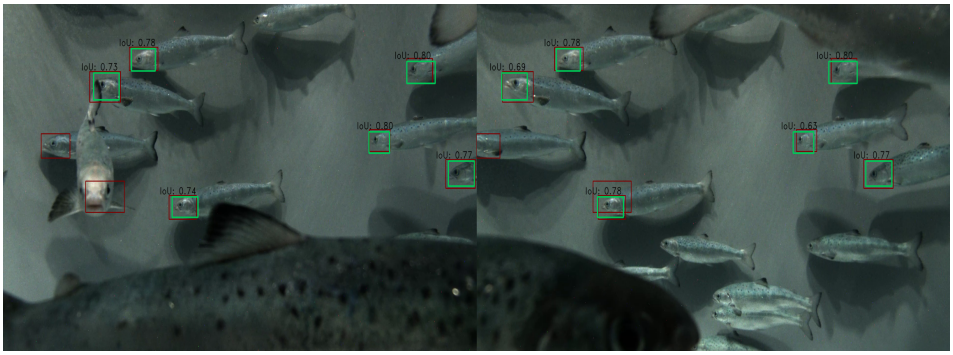


Figure 4.4: *Example test image 2*: Detections in a stereo image.

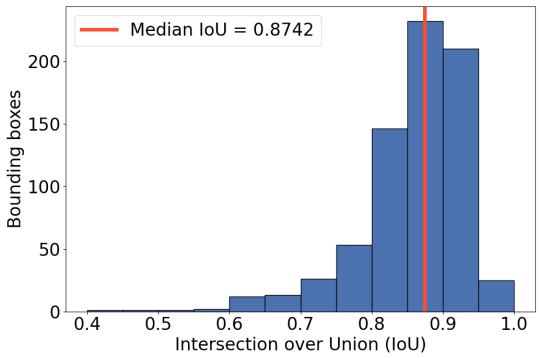


Figure 4.5: IoU distribution of the test dataset with Stereo R-CNN trained without data augmentation at epoch 50.

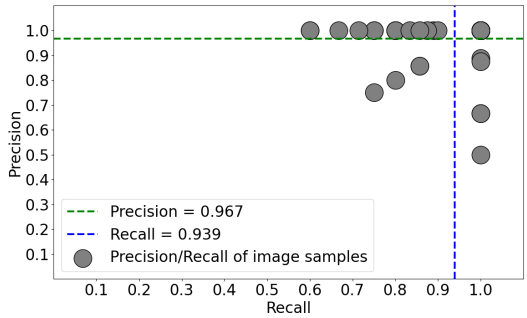


Figure 4.6: Precision and recall overview of the test dataset with Stereo R-CNN trained without data augmentation at epoch 50.

4.2 Estimated 3D Coordinates of Detected Objects

This section presents a sample of the results obtained from 3D coordinate estimation of detected objects. The same 10 images used during the Specialization Project were utilized. Three test images are presented within this section, two previously displayed in Section 4.3 in the Specialization Project [14]. This applies to *test image 2* and *3*. The remaining seven images are in Appendix C.

Test Image 1

Figure 4.7 presents the detected salmon juveniles of a stereo image pair in the test dataset. The Stereo R-CNN model detected 9 objects. Figure 4.8 illustrates the corresponding estimated x- and y-coordinates for these objects, while Figure 4.9 provides information on the z-coordinate, depth, of the objects. The $x/y/z$-coordinates only show 8 objects, excluding *object 9*. This implies that *object 9* had an unrealistic depth estimate and was filtered out due to depth correction introduced during the Specialization Project. For more information on the subject, see Subsection 3.7.1 in Specialization Project [14]. The other detected objects in the image indicated satisfactory results, suggesting *object 6* closest to the camera, and *object 2* and *3* furthest away.

Figure 4.7: *Test image 1*: Detected salmon juveniles in stereo image.

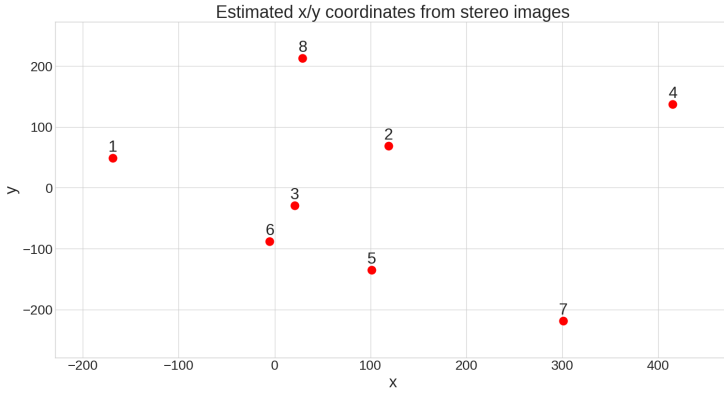


Figure 4.8: *Test image 1*: Estimated x- and y-position of salmon juveniles in stereo image.

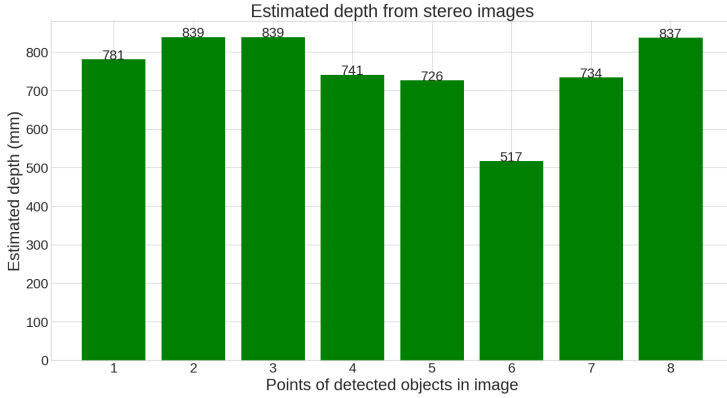


Figure 4.9: *Test image 1*: Estimated depth of salmon juveniles in stereo image.

Test Image 2

Figure 4.10 presents 5 correctly detected fish in another stereo image pair. The model employed in the Specialization Project detected 6, including one FP. False detections often led to unrealistic 3D coordinates; an improved detection model contributed to more accurate 3D coordinate estimations. Examining the associated x/y-coordinates in Figure 4.11 and depth estimations in Figure 4.12, it appears that all 5 objects have realistic estimations.

Test Image 3

Figure 4.13 demonstrates the third example of estimated 3D coordinates of detected objects within a stereo image. In the Specialization Project, the stereo image pair showed that two of the detected objects in the left image overlapped in the right image, resulting in imprecise 3D estimations for these objects. This issue did not occur utilizing the new model. Figure 4.14 presents the x/y-coordinates of the detected objects, indicating that *object 5* is located further to the left than *object 8*. Upon comparing these estimates with the stereo image, they appear to be inaccurate. Figure 4.15 displays the estimated depth, positioning *object 8* closest to the camera. This proximity could account for the discrepancy in the x/y estimations, given that the coordinate system defined by the camera calibration might differ from what the eye observes from the stereo image.

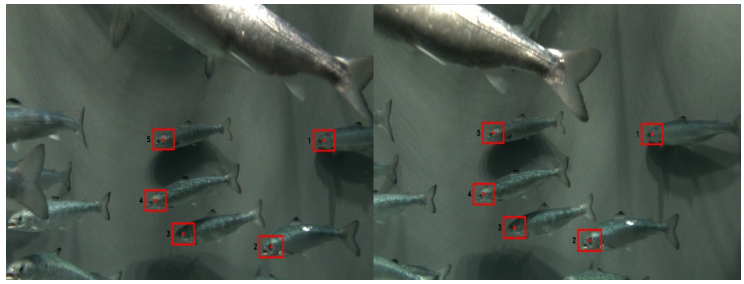


Figure 4.10: *Test image 2*: Detected salmon juveniles in stereo image.

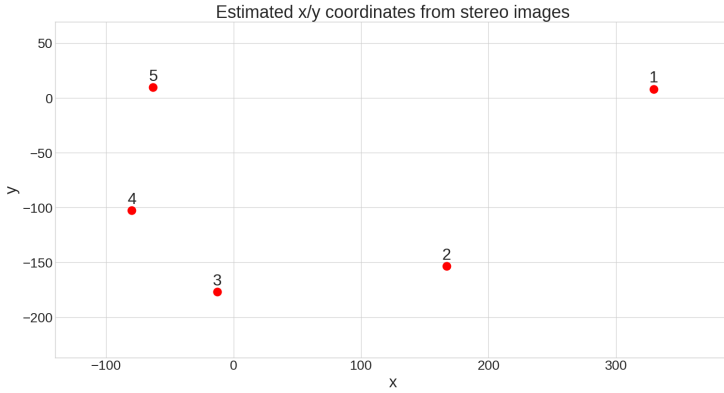


Figure 4.11: *Test image 2*: Estimated x- and y-position of salmon juveniles in stereo image.

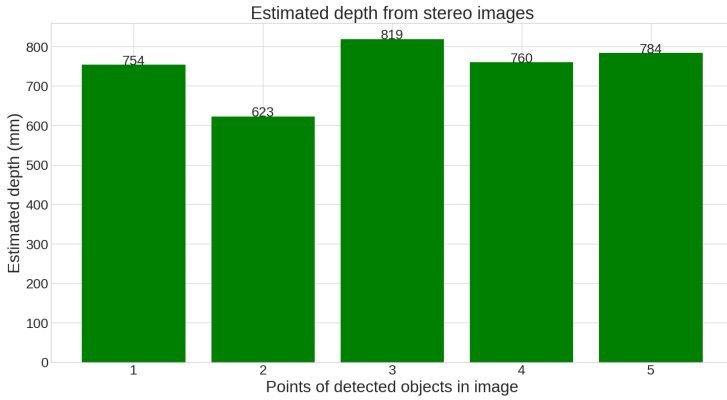


Figure 4.12: *Test image 2*: Estimated depth of salmon juveniles in stereo image.

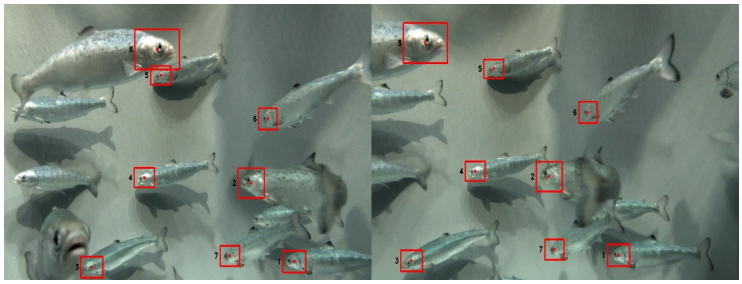


Figure 4.13: *Test image 3*: Detected salmon juveniles in stereo image.

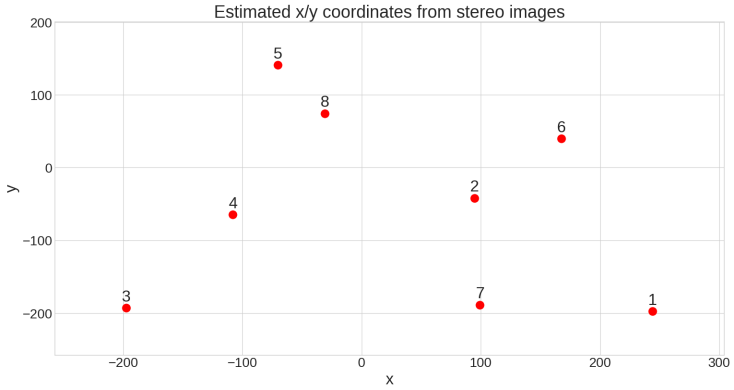


Figure 4.14: *Test image 3*: Estimated x- and y-position of salmon juveniles in stereo image.

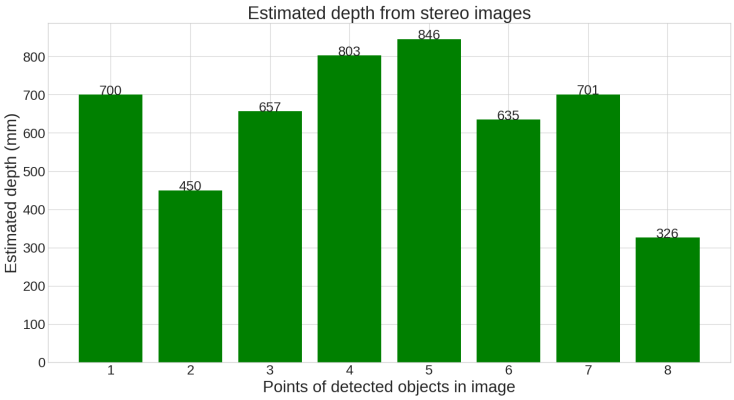


Figure 4.15: *Test image 3*: Estimated depth of salmon juveniles in stereo image.

4.3 Object Tracking

This section presents the obtained results from object-tracking salmon juveniles in videos. All provided videos from SINTEF Ocean were tracked to retrieve positional data. The results from experimenting with the T_{lost} and frame count values will be presented. Figure 4.16 is a snapshot of an object-tracked video. Two examples of object-tracked videos are provided in Supplementary Materials. The two videos present object tracking of *normal behavior* from *27.06.2022* and *stressed behavior* from *07.07.2022*.

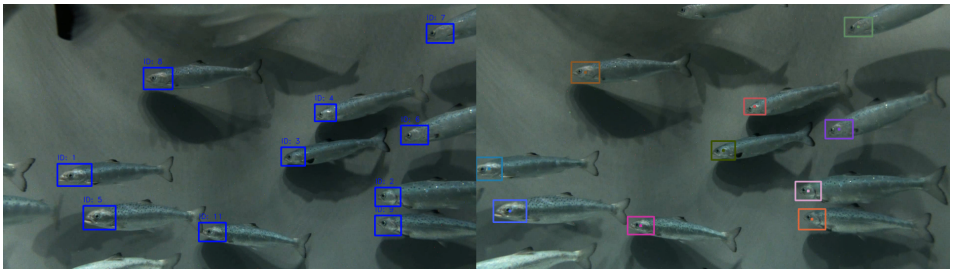


Figure 4.16: Snapshot of an object-tracked salmon juvenile video.

4.3.1 Time Series of Positional Data

A 5-minute video from *27.06.2022* was utilized to test different values of T_{lost} and frame count for object tracking. Table 4.2 presents an overview of the obtained number of detected fish and average time series per fish from the combinations of T_{lost} and frame count. The experimental outcomes indicated best results with $T_{lost} = 5$ and $f_c = 15$ frames.

T_{lost} Threshold

Three different T_{lost} thresholds were tested to acquire extended time series of positional data. The frame count was kept constant with a value of $f_c = 15$. An example of a fish's positional data with T_{lost} set to 1, 5, and 20 can be seen in Figure 4.17, 4.18, and 4.19, respectively. The time series are from the same fish. The average time series per fish presented in Table 4.2 indicates that the lower the value of T_{lost}, the shorter the time series for fish. A smaller T_{lost} value caused the SORT algorithm to remove tracks more quickly when no matching detections were found, while a larger value allowed the algorithm to retain tracks for a longer

period. For two other examples of time series of a fish with various T_{lost} values, see Appendix D.

Table 4.2: Overview of tested values for T_{lost} and frame count, the corresponding number of detected fish, and average time series per fish.

Threshold	Frame count	Detected fish	Average time series per fish
$T_{lost} = 1$	15 frames	839	3.384 sec
$T_{lost} = 5$	5 frames	583	3.393 sec
$T_{lost} = 5$	15 frames	601	3.661 sec
$T_{lost} = 5$	30 frames	578	3.596 sec
$T_{lost} = 10$	15 frames	530	4.024 sec
$T_{lost} = 20$	15 frames	484	4.636 sec

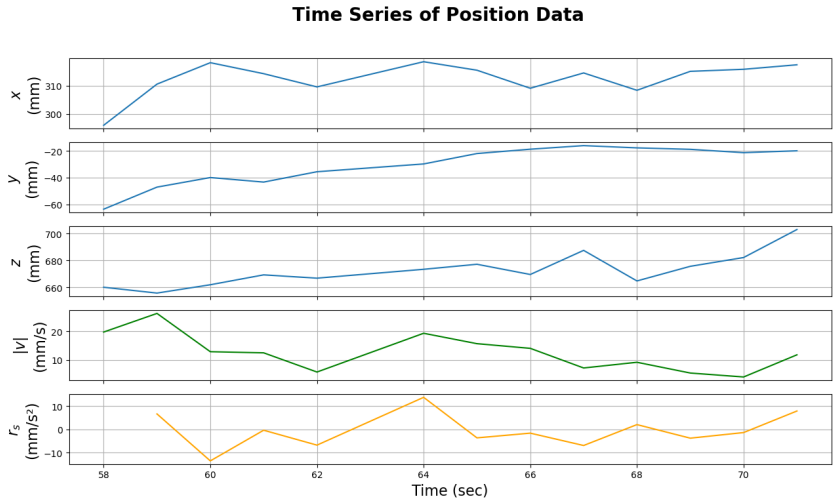


Figure 4.17: $T_{lost} = 1$, $f_c = 15$ frames: Example of time series of positional data of a fish.

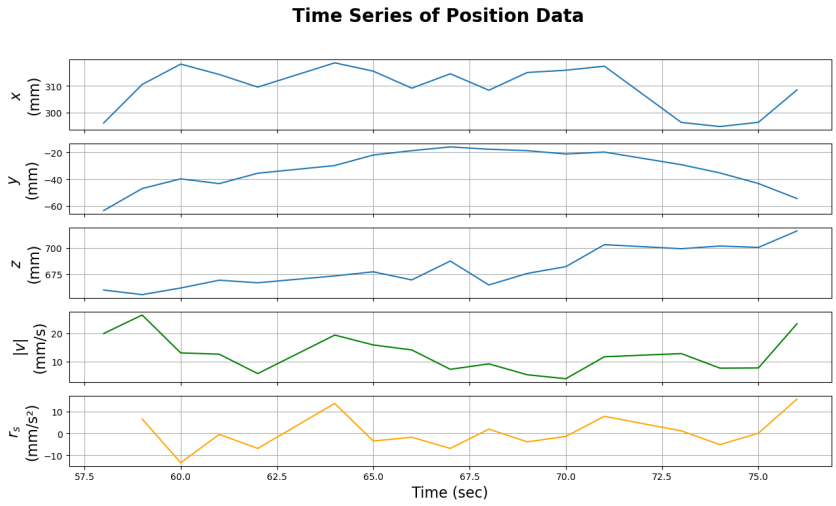


Figure 4.18: $T_{lost} = 5$, $f_c = 15$ frames: Example of time series of positional data of a fish.

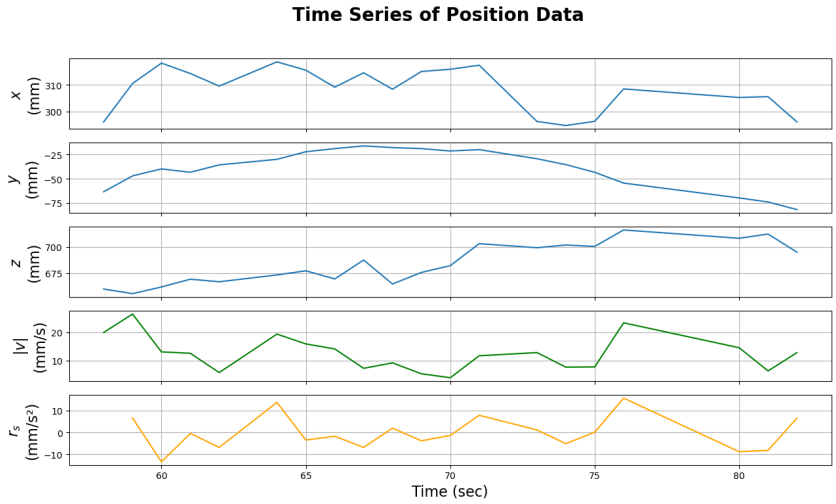


Figure 4.19: $T_{lost} = 20$, $f_c = 15$ frames: Example of time series of positional data of a fish.

Frame Count

The parameter frame count was also modified while the threshold value $T_{lost} = 5$ was kept constant to find the optimal balance between accuracy and responsiveness. The frame count was set to 5, 15, and 30 frames, and an example of positional time series obtained from every count is presented in Figure 4.20, 4.21, and 4.22, respectively. The time series are from the same fish and illustrate how variations in the quantity of historical data can influence the positional estimations. $f_c = 5$ gave noisy positional data, which resulted in large values for velocity and speed change rate. A larger frame count of 15 and 30 appeared more effective in smoothing out the variations in the data. However, a frame count of 30 gave unrealistic small velocity and speed change rate values. For two additional examples of time series with various frame counts, see Appendix D.

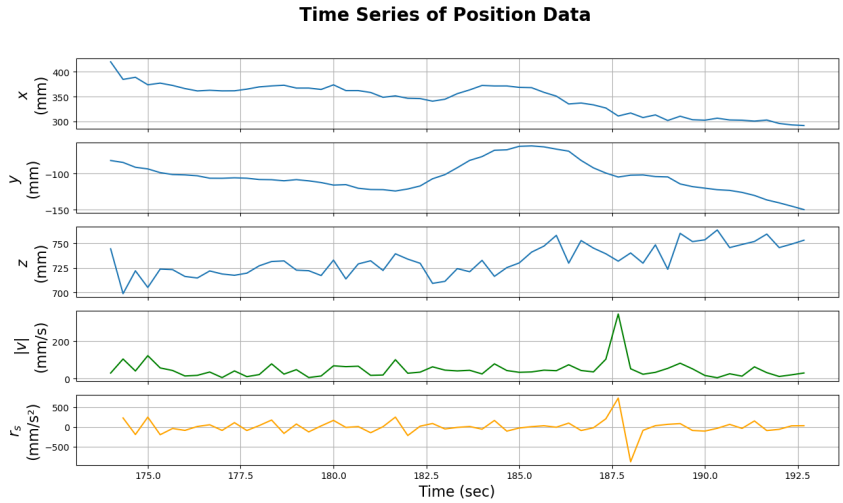


Figure 4.20: $T_{lost} = 5$, $f_c = 5$ frames: Example of time series of positional data of a fish.

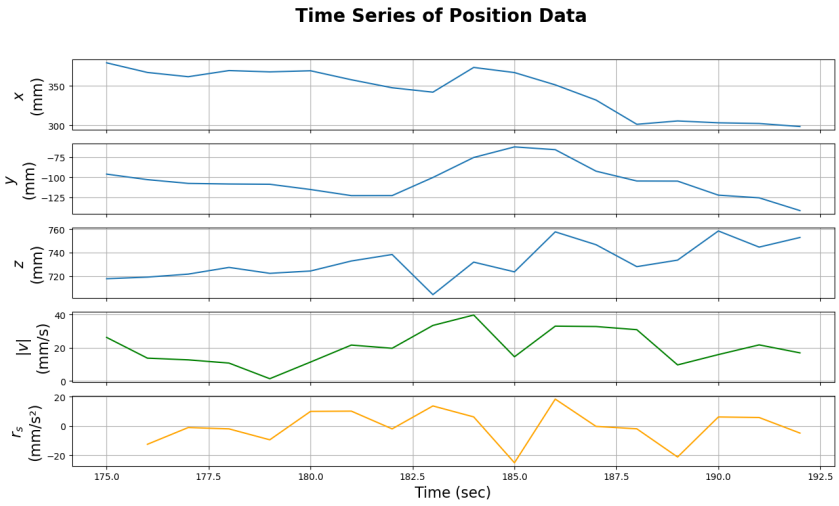


Figure 4.21: T_{lost} = 5, f_c = 15 frames: Example of time series of positional data of a fish.

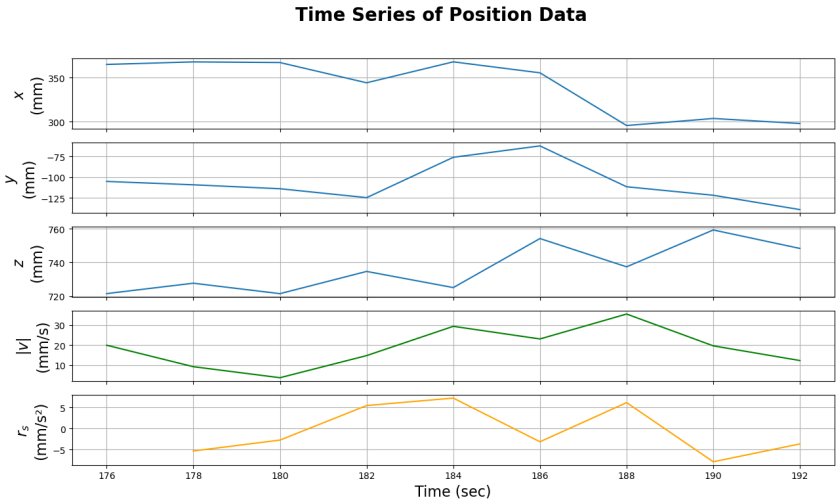


Figure 4.22: T_{lost} = 5, f_c = 30 frames: Example of time series of positional data of a fish.

4.4 H_2S Detection using Machine Learning

This section presents the results obtained from classification and estimation of H_2S concentration. Each model was evaluated on the different datasets, and the corresponding performance metrics are presented in tables. Notably, the best performance achieved by each model is highlighted in blue.

4.4.1 Classification of H_2S Concentration

This section presents the obtained results from the classification of H_2S concentration. Three support vector classifiers, a decision tree, a random forest, and an automated machine learning model were utilized to obtain the following results.

Support Vector Classifier

Table 4.3 presents an overview of achieved accuracy with the SVC utilizing different kernels on the datasets. The best accuracy was achieved with the linear kernel, reaching an accuracy of 50% with the dataset created with $w_s = 5$ minutes, $s_s = 2.5$ minutes, and $b_n = 50$. The resulting confusion matrix for this model is presented in Figure 4.23. It correctly classified some lower H_2S concentrations but misclassified other low values as the highest concentration of $67.7\mu g/L$.

The highest accuracy achieved with the RBF kernel was 28.6%, and the confusion matrix is illustrated in Figure 4.24. This model classified all samples in the test dataset as $0\mu g/L$, except for one correctly classified sample of $66.4\mu g/L$. This indicated that the RBF kernel might not be efficient at classifying positional distributions.

The polynomial kernel achieved an accuracy of 31.2%. Figure 4.25 presents the resulting confusion matrix. The model only estimated three concentrations of 0, 7.2, and $66.4\mu g/L$, where most were wrongly classified. These results suggested poor estimation with the polynomial kernel.

Table 4.3: Accuracy of SVC with different kernel functions trained on different datasets.

Accuracy of SVC with different kernel functions					
Window size	**Step size**	**Number of bins**	**Linear kernel**	**RBF kernel**	**Polynomial kernel**
5 min	2.5 min	50	50%	25%	31.2%
5 min	2.5 min	100	18.8%	18.8%	25%
10 min	2.5 min	50	42.9%	28.6%	28.6%
10 min	5 min	50	40%	26.7%	20%
10 min	5 min	100	44.4%	22.2%	27.8%

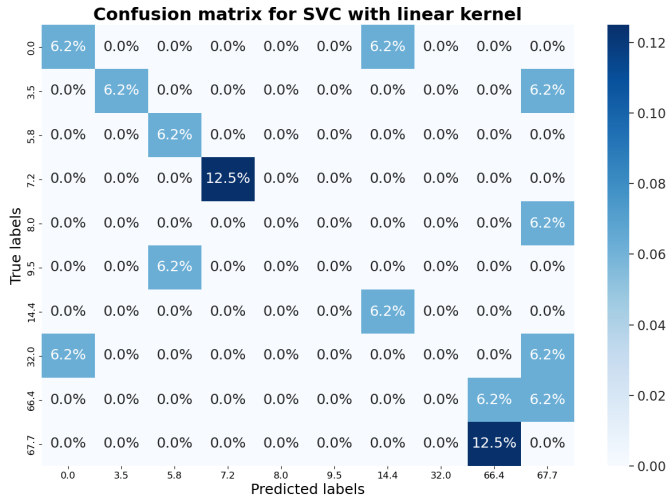


Figure 4.23: Confusion matrix for SVC with linear kernel with accuracy of 50%, trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 50$.

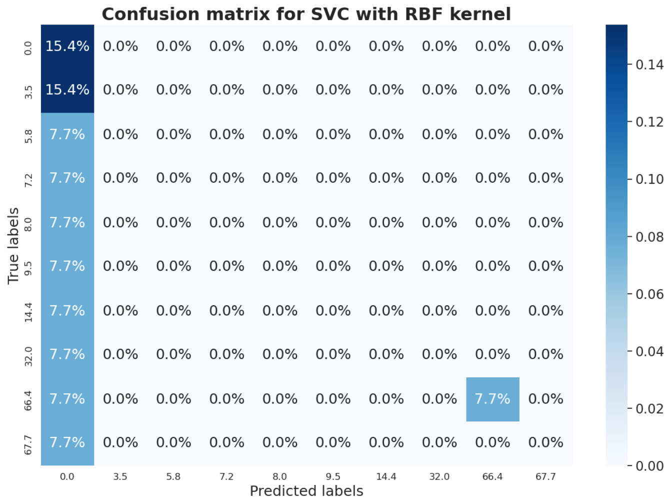


Figure 4.24: Confusion matrix for SVC with RBF kernel with accuracy of 28.6%, trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

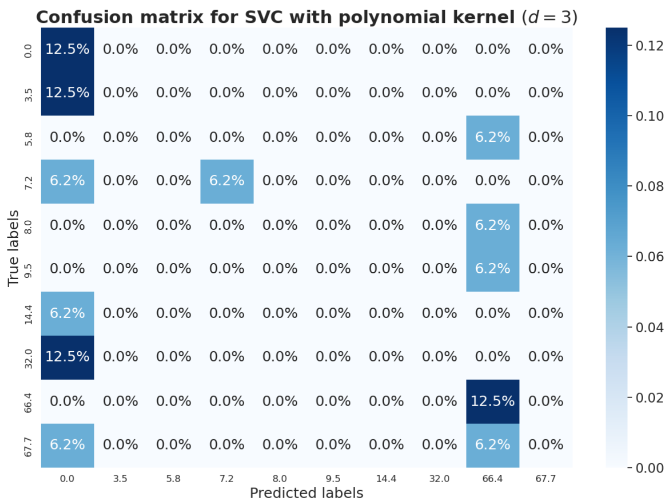


Figure 4.25: Confusion matrix for SVC with polynomial kernel with accuracy of 31.2%, trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 50$.

Decision Tree Classifier

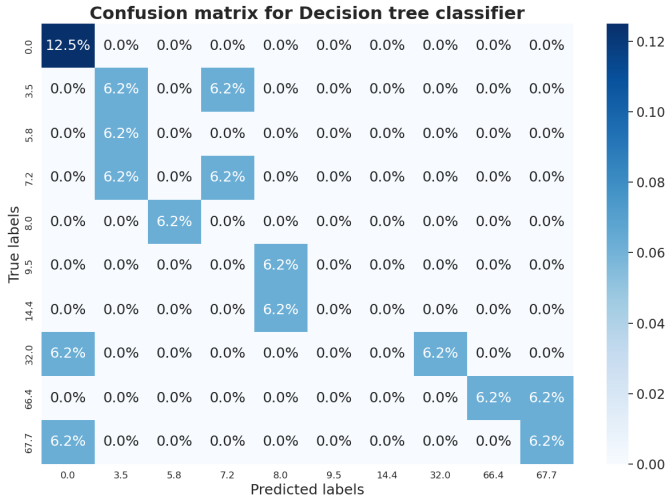
Table 4.4 presents accuracies of the decision tree dependent on the different datasets. The highest accuracy was 37.5%. This was achieved with the datasets made with $w_s = 5$ minutes, $s_s = 2.5$ minutes, and $b_n = 50$ and 100. Figure 4.26a presents the confusion matrix for the decision tree with the mentioned values and $b_n = 50$. This matrix indicates that most concentrations over $60\mu g/L$ were correctly classified as high concentrations, except for one.

Figure 4.26b presents the confusion matrix for $w_s = 10$ minutes, $s_s = 2.5$ minutes, and $b_n = 50$. By comparison of the before mentioned model, all high concentrations were classified as high concentrations. Additionally, it classified one $0\mu g/L$ as $67.7\mu g/L$.

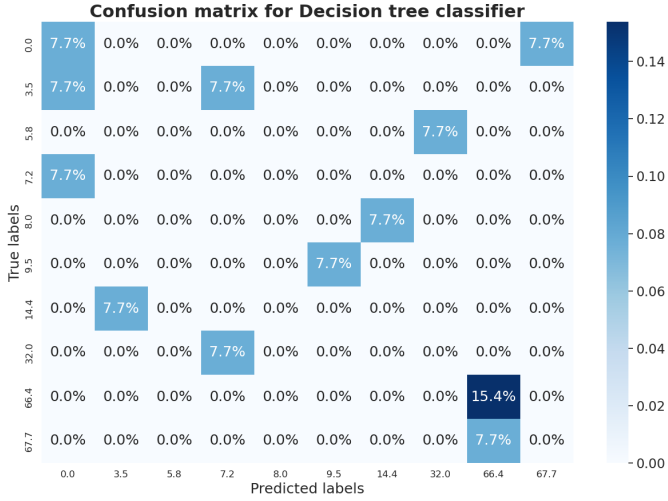
Depending on specific requirements, a model with a lower accuracy could be considered as effective as a model with higher accuracy. There seemed to be a trade-off between low values being misclassified as high and high values as low. It might be more critical to detect high concentrations rather than low concentrations accurately. Hence, the model with an accuracy of 30.8%, correctly classifying all high concentrations, may be deemed more effective when compared to the model with an accuracy of 37.5%, which misclassified some high concentrations.

Table 4.4: Accuracy of decision tree classifier trained on different datasets.

Accuracy of decision tree classifier			
Window size	**Step size**	**Number of bins**	**Accuracy**
5 min	2.5 min	50	37.5%
5 min	2.5 min	100	37.5%
10 min	2.5 min	50	30.8%
10 min	5 min	50	23.1%
10 min	5 min	100	38.5%



(a) Confusion matrix for decision tree classifier with accuracy of 37.5%, trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 50$.



(b) Confusion matrix for decision tree classifier with accuracy of 30.8%, trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Figure 4.26: Confusion matrices for decision tree classifier with accuracy of 37.5% (a) and 30.8% (b).

Random Forest Classifier

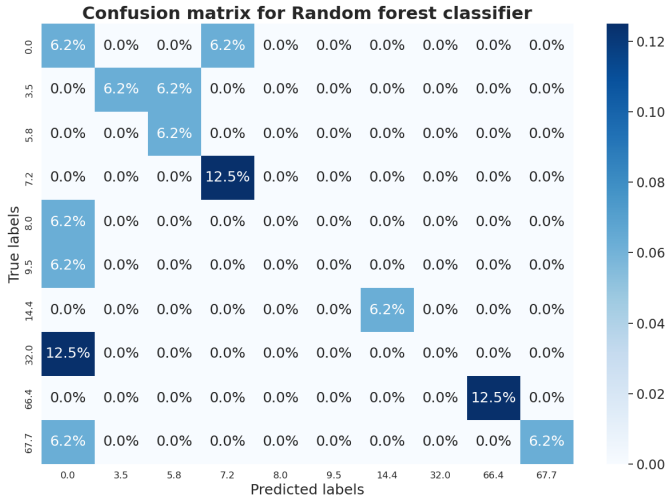
Random forest achieved the highest accuracy of all implemented classification models. Similar to the decision tree model, the highest accuracy was achieved with the datasets made with $w_s = 5$ minutes, $s_s = 2.5$ minutes, $b_n = 50$ and 100, and was 56.3%. Table 4.5 presents an accuracy overview dependent on the datasets.

Figure 4.27a presents the confusion matrix for $w_s = 5$ minutes, $s_s = 2.5$ minutes, and $b_n = 50$. The model estimated all high values as high except for one. It also classified many of the low values as too low. This trend indicates that the model often classified concentrations lower than their actual values.

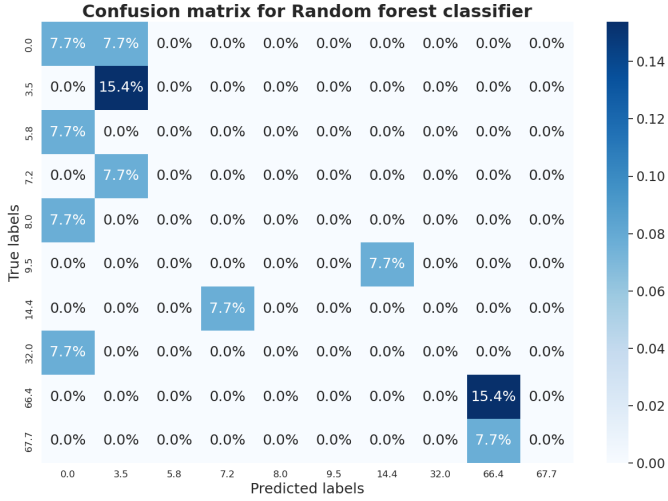
Figure 4.27b presents the confusion matrix for the accuracy of 38.5%. This model classified the highest values correctly but struggled with the lower concentrations.

Table 4.5: Accuracy of random forest classifier trained on different datasets.

Accuracy of random forest classifier			
Window size	**Step size**	**Number of bins**	**Accuracy**
5 min	2.5 min	50	56.3%
5 min	2.5 min	100	56.3%
10 min	2.5 min	50	38.5%
10 min	5 min	50	30.8%
10 min	5 min	100	30.8%



(a) Confusion matrix for random forest classifier with accuracy of 56.3%, trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 50$.



(b) Confusion matrix for random forest classifier with accuracy of 38.5%, trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Figure 4.27: Confusion matrices for random forest classifier with accuracy of 56.3% (a) and 38.5% (b).

Random Forest with Auto-Sklearn Classification

The Auto-Sklearn classification was employed to evaluate if an automatic machine learning method could outperform the other traditional models. The accuracy of the best-performing model on the different datasets is shown in Table 4.6. The best accuracy of 50% was achieved with a random forest trained with the dataset created with $w_s = 5$ minutes, $s_s = 2.5$ minutes, and $b_n = 100$. Table 4.7 presents the leaderboard of trained models of this dataset. Figure 4.28 illustrates the confusion matrix for the top-performing model. The confusion matrix reveals that three samples were accurately classified as high H_2S concentrations. However, there are many misclassifications: one sample with a high concentration was erroneously classified as $0\mu g/L$, and another sample with a true label of $0\mu g/L$ was misclassified as $67.7\mu g/L$.

Table 4.6: Accuracy of Auto-Sklearn classification trained on different datasets.

Accuracy of Auto-Sklearn classification			
Window size	**Step size**	**Number of bins**	**Accuracy**
5 min	2.5 min	50	37.5%
5 min	2.5 min	100	50.0%
10 min	2.5 min	50	30.8%
10 min	5 min	50	46.2%
10 min	5 min	100	38.5%

Table 4.7: The Auto-Sklearn leaderboard for classification models trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 100$.

Auto-Sklearn leaderboard for classification				
Rank	**Ensemble weight**	**Type**	**Cost**	**Duration**
1	0.18	Random forest	0.215	2.755
2	0.12	Random forest	0.228	2.020
3	0.02	Random forest	0.228	1.597
4	0.18	Random forest	0.241	1.974
5	0.06	Random forest	0.241	1.804

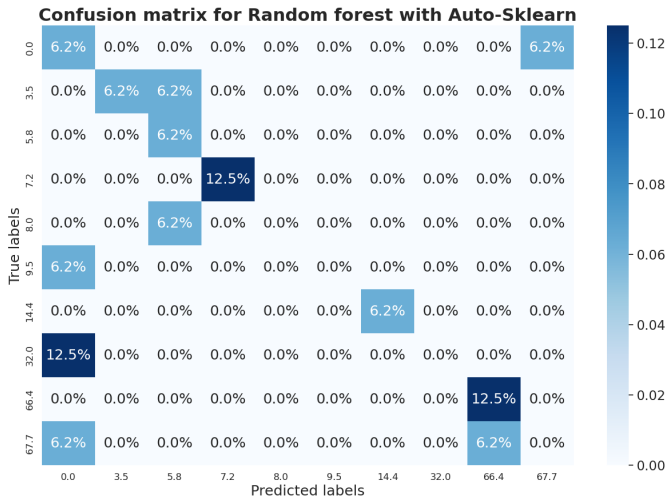


Figure 4.28: Confusion matrix for random forest provided by Auto-Sklearn with accuracy of 50%, trained with the dataset: $w_s = 5$ min, $s_s = 2.5$ min, and $b_n = 100$.

4.4.2 Estimation of H_2S Concentration

This section presents the achieved results from the estimation of H_2S concentration. Five machine learning methods were employed; a support vector regressor, a decision tree, a random forest, and two automatic machine learning methods. Their performance was evaluated using the five distinct datasets.

Support Vector Regressor

The first machine learning method tested was SVR with RBF kernel. Table 4.8 presents the R^2 score, MAE, and RMSE values for the different datasets. The performance of the SVR model was unsatisfactory, with the highest R^2 score achieved being -0.342. This negative value indicates that the model failed to capture the underlying patterns in the data and that the model's estimations were worse than a constant function that always estimates the mean of the data. Figure 4.29 presents the regression plot comparing the estimated values to the true values. A significant deviation between the estimated and true values is evident, as exemplified by the three rightmost data points. In these instances, the model estimated lower values than $8.5\mu g/L$, while the true values were over $60\mu g/L$.

Table 4.8: Performance of SVR with RBF kernel trained on different datasets.

Performance of SVR with RBF kernel					
Window size	**Step size**	**Number of bins**	**R² score**	**MAE**	**RMSE**
5 min	2.5 min	50	-0.440	20.354	31.447
5 min	2.5 min	100	-0.440	20.353	31.443
10 min	2.5 min	50	-0.342	18.467	29.920
10 min	5 min	50	-0.346	18.341	29.973
10 min	5 min	100	-0.346	18.339	29.969

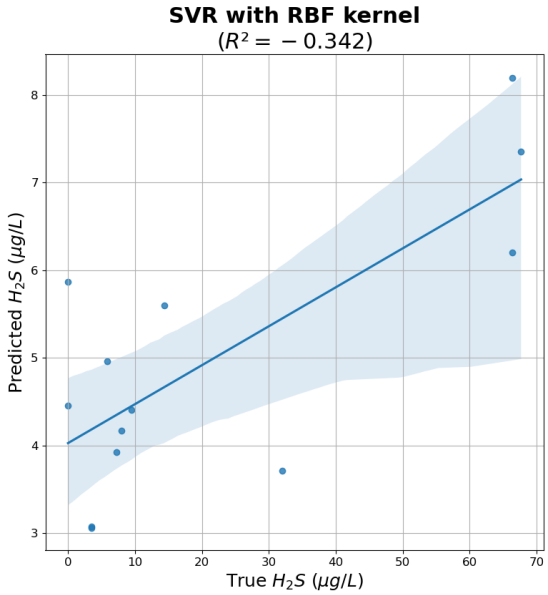


Figure 4.29: Regression plot for SVR with RBF kernel trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Decision Tree Regressor

The second method evaluated was decision tree regression. The model's performance was significantly better than the SVR, with the highest achieved R² score of 0.856 with the dataset created with $w_s = 10$ minutes, $s_s = 5$ minutes, and $b_n = 50$. Table 4.9 presents the other model performances.

Figure 4.30 presents the corresponding regression plot for the best-performing model with an R^2 score of 0.856, MAE of 5.392, and RMSE of 9.816. These metrics indicate that the model is generally effective in estimating H_2S concentrations, explaining approximately 85.6% of the variance in the dataset. The model's average prediction error, RMSE, of 9.816 means that the predicted values deviate from the true values by approximately $9.816 \mu g/L$. A closer examination of the plot reveals that the confidence intervals are not evenly distributed across the entire range of true values. The confidence intervals are relatively narrow for the lower part of the plot, suggesting that the model provides precise estimations for lower H_2S concentrations. On the other hand, the confidence intervals become considerably wider as the true values increase, indicating that the model's estimate performance decreases for higher H_2S concentrations. This deviation appears to be primarily caused by a single data point with a true value of $32 \mu g/L$, which the model estimated to be $0 \mu g/L$.

Table 4.9: Performance of decision tree regressor trained on different datasets.

Performance of decision tree regressor					
Window size	**Step size**	**Number of bins**	**R^2 score**	**MAE**	**RMSE**
5 min	2.5 min	50	0.038	14.356	25.694
5 min	2.5 min	100	0.561	6.869	17.367
10 min	2.5 min	50	0.434	10.223	19.435
10 min	5 min	50	0.856	5.392	9.816
10 min	5 min	100	0.360	9.508	20.670

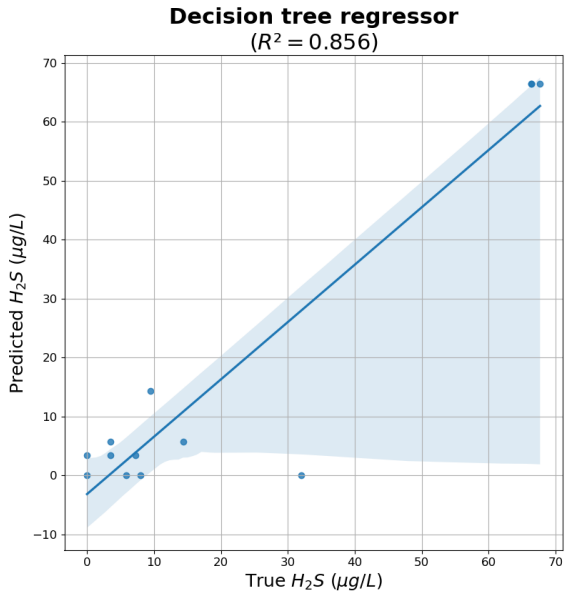


Figure 4.30: Regression plot for decision tree regressor trained with the dataset: $w_s = 10$ min, $s_s = 5$ min, and $b_n = 50$.

Random Forest Regressor

Random forest was the third method tested for H_2S estimation. This ensemble method displayed a slightly bigger improvement compared to the decision tree. The highest R² score was 0.9, with an MAE of 4.612 and an RMSE of 8.174. Table 4.10 presents the results of the model trained on different datasets.

An R² score of 0.9 suggests that the model was able to explain about 90% of the variance in the H_2S concentration data. Figure 4.31 illustrates the regression plot of this model. Similar characteristics to those observed in the decision tree plot are also evident. A comparative analysis reveals that the estimations for lower values were more accurate. Additionally, the dispersion in the confidence intervals appear to be predominantly influenced by the same single data point with the true value of $32\mu g/L$. The model estimated this particular value to be approximately $7\mu g/L$.

Table 4.10: Performance of random forest regressor trained on different datasets.

Performance of random forest regressor					
Window size	**Step size**	**Number of bins**	**R² score**	**MAE**	**RMSE**
5 min	2.5 min	50	0.526	9.866	18.035
5 min	2.5 min	100	0.527	9.761	18.018
10 min	2.5 min	50	0.900	4.612	8.174
10 min	5 min	50	0.880	5.526	8.924
10 min	5 min	100	0.531	9.844	17.687

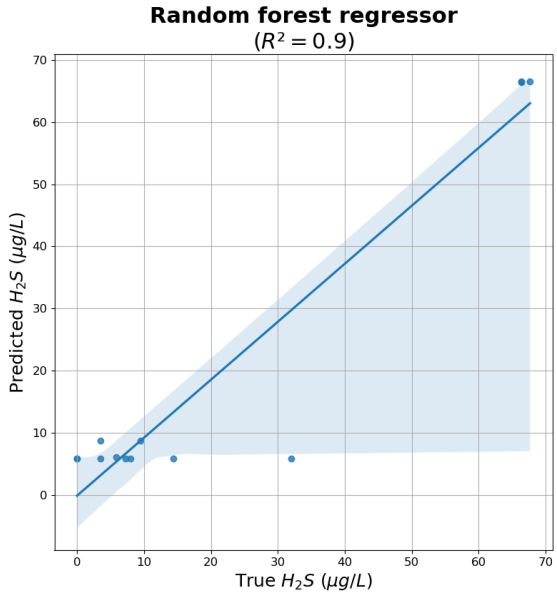


Figure 4.31: Regression plot for random forest regressor trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Decision Tree with Auto-Sklearn Regression

The Auto-Sklearn regression identified a decision tree model as the top-performing model, with an R² score of 0.869, an MAE of 6.422, and an RMSE of 9.34. This was achieved with the dataset created with $w_s = 10$ minutes, $s_s = 2.5$ minutes, and $b_n = 50$. The performance metrics indicate that the decision tree model per-

formed well but slightly worse than the previously tested random forest regressor. The other metrics for the different datasets are presented in Table 4.11. Table 4.12 shows the leaderboard of Auto-Sklearn of the best dataset.

Figure 4.32 illustrates the regression plot of the top model. A closer examination of the regression plot indicates smaller variance for higher H_2S concentrations than the previous models. Notably, the data point with a true value of $32\mu g/L$ was estimated to be $10\mu g/L$, which was closer to the true value. Additionally, the higher and lower H_2S concentration values appear to be more spread, which suggests that the model may provide a more balanced performance across the entire range of concentrations.

Table 4.11: Performance of Auto-Sklearn regression trained on different datasets.

Performance of Auto-Sklearn regression					
Window size	**Step size**	**Number of bins**	**R² score**	**MAE**	**RMSE**
5 min	2.5 min	50	0.517	10.825	18.217
5 min	2.5 min	100	0.534	9.646	17.895
10 min	2.5 min	50	0.869	6.422	9.340
10 min	5 min	50	0.609	10.994	16.154
10 min	5 min	100	0.489	13.405	18.466

Table 4.12: The Auto-Sklearn leaderboard for regression models trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Auto-Sklearn leaderboard for regression				
Rank	**Ensemble weight**	**Type**	**Cost**	**Duration**
1	0.62	Decision tree	0.064	0.292
2	0.10	Extra trees	0.099	12.743
3	0.02	Adaboost	0.156	1.243
4	0.04	Gaussian process	0.188	43.210
5	0.04	Gradient boosting	0.208	6.831

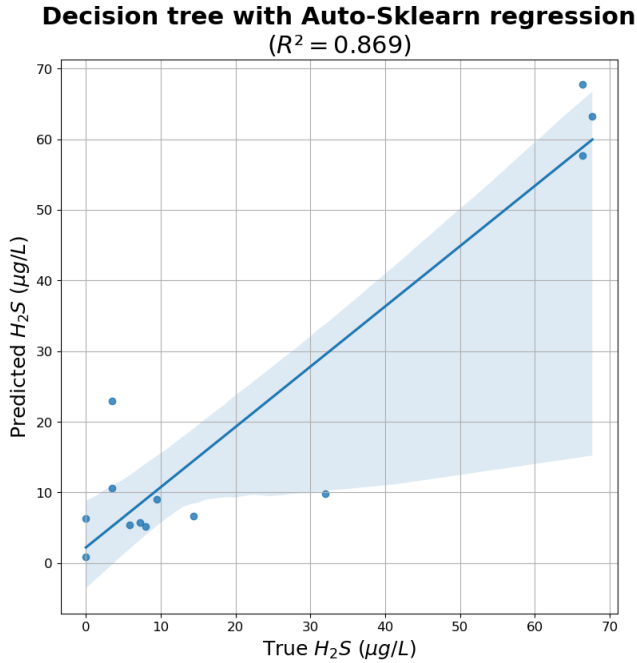


Figure 4.32: Regression plot for decision tree provided by Auto-Sklearn trained with the dataset: $w_s = 10$ min, $s_s = 2.5$ min, and $b_n = 50$.

Stack Ensemble Model with H2O AutoML

Table 4.13 presents an overview of the performance of the utilized H2O AutoML method on the different datasets. A stacked ensemble model was identified as the top-performing model, achieved with the dataset of $w_s = 10$ minutes, $s_s = 5$ minutes, and $b_n = 50$. The corresponding performance metrics were an R² score of 0.887, an MAE of 4.756, and an RMSE of 8.677, indicating that the model performed well in estimating H_2S concentrations. The specifications of this model are provided in Table 4.14, followed by an explanation.

Figure 4.33 illustrates the regression plot of the stacked ensemble model. The narrower variance observed in the model's estimations was notably improved over the previously tested methods. Specifically, the higher variance interval for the stacked ensemble model ranges from approximately 30 to $70 \mu g/L$. This result reveals that the stacked ensemble model offered a more accurate and consistent estimation performance when estimating H_2S concentrations of higher values.

Table 4.13: Performance of H2O AutoML regression trained on different datasets.

Performance of H2O AutoML regression					
Window size	**Step size**	**Number of bins**	**R^2 score**	**MAE**	**RMSE**
5 min	2.5 min	50	0.462	10.615	19.221
5 min	2.5 min	100	0.395	12.890	20.377
10 min	2.5 min	50	0.760	8.335	12.659
10 min	5 min	50	0.887	4.756	8.677
10 min	5 min	100	0.485	9.423	18.537

Table 4.14: Model summary of the top performing stacked ensemble model provided by H20 AutoML, trained with the dataset: $w_s = 10$ min, $s_s = 5$ min, and $b_n = 50$.

Model summary for stacked ensemble	
Key	**Value**
Stacking strategy	cross validation
Number of base models (used / total)	3/6
GBM base models (used / total)	1/1
XGBoost base models (used / total)	1/1
DRF base models (used / total)	0/2
DeepLearning base models (used / total)	1/1
GLM base models (used / total)	0/1
Metalearner algorithm	GLM
Metalearner fold assignment scheme	Random
Metalearner nfolds	5
Metalearner fold_column	None
Custom metalearner hyperparameters	None

Explanation of the top performing stacked ensemble model presented in Table 4.14[1]:

- **Stacking strategy**: Cross-validation was used to combine the base models, meaning that cross-validated base model predictions are used to train the metalearner.
- **Number of base models (used / total)**: Indicates that 3 out of the total 6 base models were selected to be included in the stacked ensemble.
- **GBM base models (used / total)**: 1 Gradient Boosting Machine (GBM) model was generated and included in the ensemble.
- **XGBoost base models (used / total)**: 1 XGBoost model was generated and included in the ensemble.
- **DRF base models (used / total)**: 2 Distributed Random Forest (DRF) models were generated, but none were included in the ensemble.
- **DeepLearning base models (used / total)**: 1 Deep Learning model was generated and included in the ensemble.
- **GLM base models (used / total)**: 1 Generalized Linear Model (GLM) was generated, but not included in the ensemble.
- **Metalearner algorithm**: The algorithm combines the base models' predictions. In this case, it is a Generalized Linear Model (GLM).
- **Metalearner fold assignment scheme**: The method used for assigning data to folds in the metalearner. "Random" indicates that data points are randomly assigned to folds.
- **Metalearner nfolds**: 5 cross-validation folds used in the metalearner training process.
- **Metalearner fold_column**: If a specific column was used to assign data to folds, its name would be mentioned here. "None" indicates that no specific column was used.
- **Custom metalearner hyperparameters**: Any custom hyperparameters provided for the metalearner would be listed here. In this case, no custom hyperparameters were used.

[1]Model explanation list rephrased from: docs.h2o.ai/stackedensembleestimator.

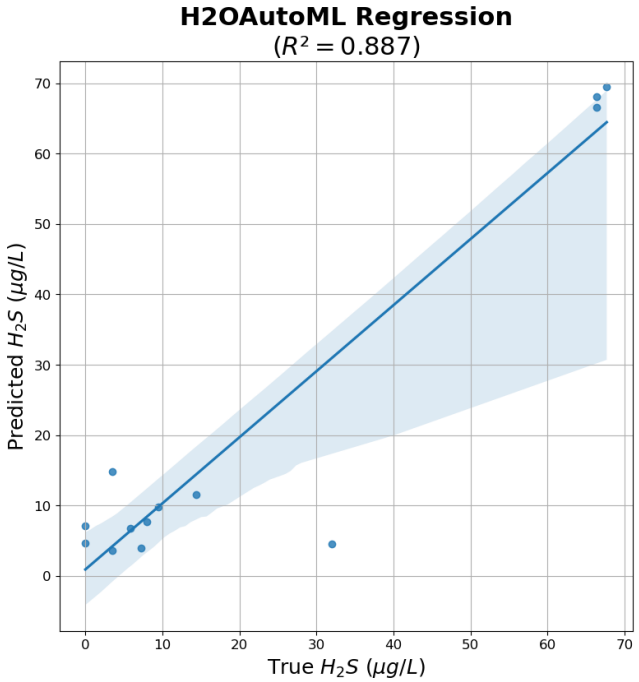


Figure 4.33: Regression plot for stacked ensemble model provided by H2O AutoML trained with the dataset: $w_s = 10$ min, $s_s = 5$ min, and $b_n = 50$.

5

Discussion

This chapter presents a comprehensive discussion of results that have emerged during the project. The discourse begins with the process of training and testing the Stereo R-CNN, followed by the 3D coordinate estimation of detected objects and object tacking. An essential part of the discussion concerns the acquired positional data. Here, an assessment of the collected data and reflections around the created datasets are provided. The last results discussed are classification and estimation of H_2S concentration.

The discussion ends with a summary that aims to weave together these diverse topics, facilitating a holistic understanding of the research project and potential use. This is followed by the project's limitations and suggested future work.

5.1 Training and Testing Stereo R-CNN

The training of the Stereo R-CNN with an expanded dataset yielded satisfactory results. The training and validation loss plot revealed a minimum loss of -21, surpassing the -17 loss from the Specialization Project, indicating a substantial improvement. The minimum was achieved by training the model without data augmentation, over approximately 50 epochs. The validation loss graph indicated overfitting beyond this point.

Given that the best-performing model was achieved without data augmentation, the larger dataset potentially led the model to reach the saturation point of performance

improvement. This implied that the expansion of the dataset brought the model's learning capacity to its optimum level, where further data or the application of data augmentation could not yield additional improvements.

Expanding the test dataset from 10 to 66 stereo images facilitated a more robust foundation for determining the optimal model at a given epoch. The best model depended on the trade-offs between median IoU, precision, and recall. The decision was mainly based on the balance between identifying as many true positives as possible (higher recall) and the accuracy of positive predictions (higher precision). The model trained without data augmentation at epoch 50 was chosen based on its high precision and recall. However, the variances in performance were minor, suggesting that many of the models trained without data augmentation could be suitable for object detection.

The top-performing model achieved a median IoU of 0.874, meaning that the detected bounding boxes and the ground truth boxes overlapped with 87.4%. The precision and recall were 96.2% and 93.9%, respectively. It can be concluded that the model was performing well, based on these metrics and observations of detected objects from testing.

5.2 Estimated 3D Coordinates of Detected Objects

The test set of 10 images presented satisfactory results compared to those obtained during the Specialization Project. As stated in the Specialization Project [14], the primary source of inaccurate 3D estimates appeared to be miss-detections by the Stereo R-CNN. Therefore, improved results were achieved using the new Stereo R-CNN model as the utilized camera calibration parameters were the same as in the Specialization Project.

Although the 3D estimates were improved, some 3D coordinates were wrongly estimated. Given the potential attainment of a saturation point by the current Stereo R-CNN model, exploring alternative models may be an approach for obtaining better 3D estimate results. In addition, refining the calibration process could also result in improvements.

5.3 Object Tracking

The SORT algorithm demonstrated good results for tracking fish through video sequences. The algorithm successfully tracked fish over time with unique IDs. The optimal values for the tracking parameters T_{lost} threshold and frame count were experimentally established by visually inspecting time series of simple fish position. A smaller T_{lost} value caused the SORT algorithm to remove tracks more quickly when no matching detections were found, leading to more detected fish. Conversely, a larger value allowed the algorithm to retain tracks longer, without matching detections. This introduced more significant deviations in positional data. This could result from the underlying constant velocity model utilized by the SORT algorithm for estimating fish positions when lost for a number of frames. By decreasing the frame count, the Kalman filter would generate more frequent updates, potentially improving its ability to track positional changes and respond to abrupt shifts in position. In contrast, increasing the frame count would yield less responsiveness to sudden position changes but produce comparatively smoother results.

The objective of obtaining longer time series of fish positional data was accomplished. By incrementing the T_{lost} to 5 while maintaining the frame count as its original value of 15 frames, more extended time series were acquired in addition to smoothing out the noise. As a result, the average time series per fish increased by 8.18%.

Expanding the time series of data offered insight into the performance of the tracking algorithm. The obtainment of longer time series demonstrated that the utilized tracking algorithm consistently maintained accuracy. This is important in dynamic systems where the tracked object changes its behavior, speed, or other characteristics. Overall, the SORT algorithm and the Kalman filter demonstrated commendable performance in object tracking.

Additionally, examining the time series of 3D coordinates was a decent method to evaluate position estimates. Some deviations from anticipated positional data were observed, indicating noise and variation in position estimates. Large deviations could lead to inaccurate outcomes and present a distorted presentation of the fish population. An option to enhance the accuracy of the tracking system's position estimates is to neglect large deviations from trending values.

5.4 Acquired Positional Data

A comprehensive dataset containing positional data was obtained from object tracking of all provided videos. Various trends in positional data were apparent across the different dosages of H_2S exposure. For a majority of the observation period, specifically *27-30.06.2022*, *01.07.2022*, and *03-05.07.2022*, the positional data exhibited minimal to no behavioral changes in response to H_2S exposure. However, noticeable deviations from typical positional data were seen on *02.07.2022*, *06.07.2022*, and *07.07.2022*.

While inaccurate positional estimates could cause these patterns, similar swimming behaviors were observed upon visually examining the videos. The limited instances of abnormal fish behavior in the video data may present a challenge in creating a robust dataset for further analysis of H_2S detection. A lack of sufficient variation in swimming patterns might not allow for effective discernment of the effects of H_2S exposure.

Numerous factors might influence the lack of reaction to H_2S exposure in fish. A potential explanation is that the fish are naturally resilient to small dosages of H_2S; hence, they continue to exhibit normal behavior. Another possibility is the familiarization or acclimatization of the fish if they have experienced exposure before. Another observation from the exposure periods was the apparent immediate calmness in the fish behavior once the concentration of H_2S peaked and started declining. This behavioral pattern could indicate relief, suggesting an adaptive response to the H_2S as it dissipated.

Acquired positional data characterized merely by velocity and speed change rate provided some information but more is needed for a comprehensive understanding of swimming behaviour. Including elements such as angular velocity and acceleration can enrich the spectrum of the data. This could provide a more holistic portrayal of fish behavior, thereby capturing a greater range of deviations.

5.4.1 Distribution Datasets

The sliding window technique was utilized to collect positional distribution data to create five different datasets. The sliding window range, specifically the start and end time, was set manually based on visually examining the H_2S graph and positional data. This approach may introduce an element of subjectivity and potential bias into the data collection process. The range of the window could significantly influence the outcome of the positional distributions. A more systematic or algo-

rithmic approach to window selection, which considers the characteristics of the H_2S and positional data, such as variability or trend, could provide a more objective and reliable analysis.

Additionally, the data collection and labeling method may present some limitations due to the inherent variability of the positional data and H_2S concentrations within the selected window. The method relied on assigning the same H_2S concentration to all collected distributions from a single day, even though the H_2S value varied within the defined time window. This approach could introduce a level of imprecision. It assumed homogeneity of fish behavior and exposure levels throughout the time frame when these factors fluctuated. The oversimplification of behavioral responses to varying exposure levels may lead to the masking of subtle but meaningful variations in swimming patterns. Future studies could benefit from exploring more dynamic labeling techniques that account for the temporal variations in H_2S concentrations and the corresponding changes in fish behavior.

Another possible limitation concerning distributional data could arise when the number of positional data points forming the distribution is less than the number of bins. This discrepancy might lead to overfitting and a sparse representation of the distributions, which could distort the interpretation of the data. A potential solution is to increase the sliding window to get more positional points and decrease the number of bins.

Another vulnerability is the relatively small dataset, while each data point consists of many features. This can lead to overfitting, where the model learns the noise in the training dataset and not the underlying patterns. Techniques such as feature selection, dimensional reduction, regularization, and collecting more data can efficiently counteract this.

Detecting H_2S concentrations based on short time series intervals may not be the most effective strategy. This approach overlooks broader trends or fluctuations over time, which could be crucial indicators of behavioral changes or H_2S levels. Instead of segmenting the data into small intervals for independent analysis, it might be more beneficial to consider extended time series or all historical data. This approach facilitates monitoring the behavioral progression over time. It could lead to more accurate and comprehensive estimation models since they would consider long-term trends or variations. Moreover, considering the trends or variations as factors over time can better reflect the real-world scenarios where changes in H_2S levels or stress are typically gradual and cumulative. Therefore, this method may provide more realistic and relevant insights into H_2S detection and stress analysis.

5.5 Classification of H_2S Concentration

An SVC with three different kernels, linear, RBF, and polynomial kernel, was chosen to conduct initial tests of classifying H_2S concentration. All kernels were tested on the five datasets. The SVC with a linear kernel achieved the highest accuracy of 50%. A closer examination of the corresponding confusion matrix showed that the model estimated many samples as the highest H_2S concentration of $67.7\mu g/L$. The performance of this model was assessed unsatisfactory.

Further, a decision tree and random forest classifier were tested. The decision tree achieved an accuracy of 37.5%. However, it correctly classified most high values as high. The best accuracy of all classification models was achieved with the random forest classifier. The accuracy was 56%. Although it estimated many instances correctly, it still classified one $67.7\mu g/L$ label as $0\mu g/L$. The assessment of the model could have been more favorable as it did not accurately detect all high concentrations, which are of greater concern due to their lethal potential compared to lower concentrations.

The random forest model achieving the highest accuracy might be due to the nature of the data. The data has complex, non-linear relationships between features, which works well with ensemble methods like the random forest. The random forest does not make assumptions about the linearity of the data and can handle high-dimensional spaces well.

To summarize, the classification models did not perform well on the task of H_2S detection based on positional distribution data. This could be due to the small dataset and the characteristics of the positional data. Given that H_2S concentration is a continuous value, using classification could result in information loss. A regression model might be a more suitable choice. Additional validation metrics, such as precision and recall, could be considered to evaluate the different models better.

5.6 Estimation of H_2S Concentration

The estimation models demonstrated a better performance in detecting H_2S concentrations based on positional distribution data compared to the classification models. An SVR with an RBF kernel, a decision tree regressor, a random forest, and two AutoML methods were assessed and tested for this task.

The random forest emerged as the top performer in terms of the R^2 score, achiev-

ing a value of 0.9, with an MAE of 4.612 and RMSE of 8.174. Despite this high performance, an examination of the regression plot indicated that the confidence intervals for higher values of H_2S were smaller for the AutoML models, suggesting improved performance in those regions.

In particular, the ensemble stacked model provided by the H2O AutoML library had an R^2 score of 0.887, with an MAE of 4.756 and an RMSE of 8.677. While these metrics may appear inferior to the random forest regressor, the AutoML model had narrower confidence intervals for higher values. It indicated that the ensemble stacked model might provide more precise and reliable estimates in scenarios of high H_2S concentrations. The AutoML method's ability to naturally combine and optimize multiple algorithms may be the reason for its smaller confidence intervals for higher values. This enables it to effectively capture the complex relationships within the data, especially at higher concentration levels where the data's variability may be more pronounced.

The ensemble stacked model demonstrates a different performance profile than the random forest. While the ensemble stacked model showed an improvement in the higher H_2S interval, the random forest regressor may appear to be the more accurate and robust choice, considering the R^2 score, MAE, and RMSE values. A larger dataset would be advisable to support these findings and further investigate the potential of these models.

5.7 System Summary and Potential Use

The H_2S detection system through salmon juvenile monitoring operates as follows: it takes video footage as an input and returns an estimation of H_2S concentration as an output. This process starts when a video is passed to the tracking module. The tracking module incorporates a pretrained Stereo R-CNN object detection model and employs the SORT algorithm and Kalman filter to derive positional data of the detected fish continuously. The resultant positional data is converted into distributional data through a sliding window technique, which is then employed to estimate H_2S concentrations. The implications of having such a system that uses video footage to estimate H_2S concentrations could significantly impact the operations of RAS facilities. Some potential outcomes include:

- **Continuous monitoring**: This system could enable continuous, real-time monitoring of H_2S concentration. This would be an improvement over manual examination and could help operators respond to changes quickly.

- **Fish health and welfare**: High concentrations of H_2S can be harmful or even lethal to fish. By detecting these levels early, operators can take action to protect the health and welfare of the fish, leading to better growth rates and lower mortality.
- **Data collection and analysis**: Over time, this system could provide a valuable dataset for analysis. This might provide a better understanding of the conditions that lead to increased H_2S and how to prevent them.
- **Immediate response**: By simultaneously monitoring the H_2S level, the system could be further developed to take immediate action in case of H_2S detection.

The primary aim of the DigiRAS project is to achieve real-time detection of H_2S. The current state of the monitoring system has yet to reach this goal, as it is not currently developed to handle real-time operations. There are also some uncertainties if the current detection model and tracking algorithm are fast enough to deal with real-time. However, the obtained results showed good potential on the task of H_2S detection.

5.8 Limitations

One limitation of this study comes from the data collection and labeling methodology. The complexity and variability in positional data and H_2S concentrations were oversimplified due to the labeling technique. This also resulted in a limited number of distinctly labeled H_2S concentrations.

5.9 Future Work

The suggested future work is summarized below.

- Outlier detection and removal of positional data acquired from object tracking.
- Include other elements of positional data, such as angular velocity and acceleration, to get a more comprehensive dataset of positional data.
- Investigate other methods to label distribution data to get a more dynamic dataset.

- Further expand the distribution dataset by investigating augmentation methods.
- Investigate other variables than distributions to represent the fish population's behavior pattern.
- Investigate other models for detecting H_2S concentration based on positional data.

6

Conclusion

One objective of this thesis was to develop an annotation tool and establish a larger training dataset for the Stereo R-CNN model. A pretrained Stereo R-CNN model was integrated into the annotation process, reducing the manual work to oversee and fine-tune the annotation. The extended dataset was used for training and testing the Stereo R-CNN. The new model provided pleasing results.

Another objective was to improve the tracking algorithm to obtain longer and more precise time series for position and velocity. This improvement provided valuable insights into the performance of the SORT algorithm and the Kalman filter in object tracking. The estimates were reasonable by comparing the time series with the corresponding video, except for some deviations. Retrieved positional data from all provided videos gave a decent representation of the fish population exhibiting various swimming patterns. Further, several distribution datasets were created using the sliding window technique with different parameters. The datasets encompassed normalized distributions of decomposed velocity, velocity magnitude, and speed rate change data.

Various techniques for detecting H_2S based on positional distribution data were assessed. Both classification and regression models were utilized for this task. The final objective of this thesis was to evaluate how well H_2S concentrations could be estimated based on positional data. The outcome of various tests with different datasets revealed that classification models were not particularly effective. In contrast, the regression models performed more proficiently, with the random forest regressor model achieving the best results. This demonstrated the potential of effectively estimating H_2S concentrations through salmon juvenile monitoring with

stereo vision and machine learning.

Bibliography

[1] Al-Azawi, M., Al-Rawy, D., Al-Jobory, S., Zaghar, D., 2019. *Stereo Vision for 3D Measurement in Robot Systems*. Journal of Engineering and Sustainable Development 17. Available from: https://www.researchgate.net/publication/331960044_Stereo_Vision_for_3D_Measurement_in_Robot_Systems (Accessed: 04.03.2023).

[2] Bagarinao, T., 1992. *Sulfide as an environmental factor and toxicant: tolerance and adaptations in aquatic organisms*. Aquatic Toxicology 24, 21–62. doi:10.1016/0166-445X(92)90015-F.

[3] Benjaminsen, C., 2021. *Fish farms moving onshore*. Available from: https://www.sintef.no/en/latest-news/2021/fish-farms-moving-onshore/ (Accessed: 09.12.2022).

[4] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B., . *Simple Online and Realtime Tracking*. volume abs/1602.00763. Available from: https://arxiv.org/abs/1602.00763 (Accessed: 28.04.2023).

[5] Bregnballe, J., 2015. *A Guide to Recirculation Aquaculture*. the Food and Agriculture Organization of the United Nations (FAO) and EUROFISH International Organisation. Available from: https://www.fao.org/3/i4626e/i4626e.pdf (Accessed: 10.01.2023).

[6] Brown, G., 2010. *Ensemble Learning*. Springer US, Boston, MA. pp. 312–320. doi:10.1007/978-0-387-30164-8_252.

[7] Challa, S., Morelande, M.R., Mušicki, D., Evans, R.J., 2011. *Fundamentals of Object Tracking*. Cambridge University Press. doi:10.1017/CBO9780511975837.

[8] ClearView, . *Stereo Vision for 3D Machine Vision Applications*. Available from: https://www.clearview-imaging.com/en/blog/stereo-vision-for-3d-machine-vision-applications (Accessed: 01.11.2022).

[9] Delua, J., 2021. *Supervised vs. Unsupervised Learning: What's the Difference?* Available from: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (Accessed: 07.12.2022).

[10] Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S., Leal-Taixé, L., 2020. *MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking*. SpringerLink. doi:10.1007/s11263-020-01393-0.

[11] Developer, N., . *CUDA Toolkit*. Available from: https://developer.nvidia.com/cuda-toolkit (Accessed: 10.01.2023).

[12] DigiRAS, . *DIGIRAS*. Available from: http://www.digiras.org/ (Accessed: 22.12.2022).

[13] Dilmegani, C., 2022. *What is Data Augmentation? Techniques, Examples Benefits*. Available from: https://research.aimultiple.com/data-augmentation/ (Accessed: 07.12.2022).

[14] Eide, L., 2023. *Behavioural Monitoring of Salmon Juveniles using Stereo Vision and Machine Learning* .

[15] El-Amir, H., Hamdy, M., 2020. *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*. Apress. doi:10.1007/978-1-4842-5349-6_9.

[16] Fawagreh, K., Gaber, M.M., Elyan, E., 2014. *Random forests: from early developments to recent advancements*. Systems Science & Control Engineering 2, 602–609. doi:10.1080/21642583.2014.956265.

[17] Ghanbari, M., Goldani, M., 2021. *Support Vector Regression Parameters Optimization using Golden Sine Algorithm and its application in stock market*. volume abs/2103.11459. Available from: https://arxiv.org/abs/2103.11459 (Accessed: 01.05.2023).

[18] Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep Learning*. MIT Press. Available from: http://www.deeplearningbook.org (Accessed: 23.01.2023).

[19] Grieg Seafood, . *Recirculating Aquaculture Systems (RAS)*. Available from: https://griegseafood.com/our-impact-recirculating-aquaculture-systems (Accessed: 01.11.2022).

[20] Gul, S., Shiriyev, J., Singhal, V., Erge, O., Temizel, C., 2021. Chapter three - advanced materials and sensors in well logging, drilling, and completion operations, in: *Sustainable Materials for Oil and Gas Applications*. Gulf Professional Publishing. volume 1 of *Advanced Materials and Sensors for the Oil and Gas Industry*, pp. 93–123. doi:10.1016/B978-0-12-824380-0.00004-9.

[21] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. *Mask R-CNN*. CoRR abs/1703.06870. Available from: http://arxiv.org/abs/1703.06870 (Accessed: 15.04.2023).

[22] Huang, S., Cai, N., Pacheco, P.P., Narandes, S., Wang, Y., Xu, W., 2018. *Applications of Support Vector Machine (SVM) Learning in Cancer Genomics*. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5822181/ (Accessed: 21.01.2023).

[23] Hutter, F., Kotthoff, L., Vanschoren, J., 2019. *Automated Machine Learning: Methods, Systems, Challenges*. Springer. doi:10.1007/978-3-030-05318-5.

[24] Juel, H.H., . *Solving the problem of hydrogen sulfide on a fish farm*. Available from: https://www.innovationnewsnetwork.com/solving-the-problem-of-hydrogen-sulfide-on-a-fish-farm/5794/ (Accessed: 01.03.2023).

[25] Learn, S., . *sklearn.svm.SVC*. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (Accessed: 21.02.2023).

[26] Li, P., Chen, X., She, S., 2019. *Stereo R-CNN based 3D Object Detection for Autonomous Driving*. CoRR abs/1902.09738. Available from: http://arxiv.org/abs/1902.09738 (Accessed: 23.01.2023).

[27] Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J., 2016. *Feature Pyramid Networks for Object Detection*. CoRR abs/1612.03144. Available from: http://arxiv.org/abs/1612.03144 (Accessed: 23.01.2023).

[28] MathWorks, . *Support Vector Machine (SVM)*. Available from: https://se.mathworks.com/discovery/support-vector-machine.html (Accessed: 21.01.2023).

[29] Misund, B., 2022. *Fiskeoppdrett*. Store Norske Leksikon. Available from: https://snl.no/fiskeoppdrett (Accessed: 17.11.2022).

[30] Mowi, 2022. *Salmon Farming Industry Handbook 2022*. Available from: https://mowi.com/wp-content/uploads/2022/07/2022-Salmon-Industry-Handbook-1.pdf (Accessed: 29.12.2022).

[31] Nedreberg, Å., 2022. *Behavioural monitoring of salmon juveniles using stereo vision and machine learning* .

[32] Rokach, L., Maimon, O., 2005. *Decision Trees*. Springer US, Boston, MA. doi:10.1007/0-387-25465-X_9.

[33] SINTEF, 2021. *Fish farms moving onshore*. Available from: https://www.sintef.no/en/latest-news/2021/fish-farms-moving-onshore/ (Accessed: 29.12.2022).

[34] Smola, A.J., Vishwanathan, S., 2008. *Introduction to Machine Learning*. Cambridge University Press. Available from: https://alex.smola.org/drafts/thebook.pdf (Accessed: 14.02.2023).

[35] Tetrauni, . *Hydrogen sulfide poisoning symptoms in fish*. Available from: https://tetrauni.com/hydrogen-sulfide-poisoning-symptoms-in-fish/ (Accessed: 01.03.2023).

[36] Yu, T., Bishop, P.L., 1998. *Stratification of microbial metabolic processes and redox potential change in an aerobic biofilm studied using microelectrodes*. Water Science and Technology 37, 195–198. doi:10.1016/S0273-1223(98)00104-8.

[37] Zhang, Z., 2021. *Camera Calibration*. Springer International Publishing, Cham. pp. 130–131. doi:10.1007/978-3-030-63416-2_164.

Appendix

A Positional Data of Fish

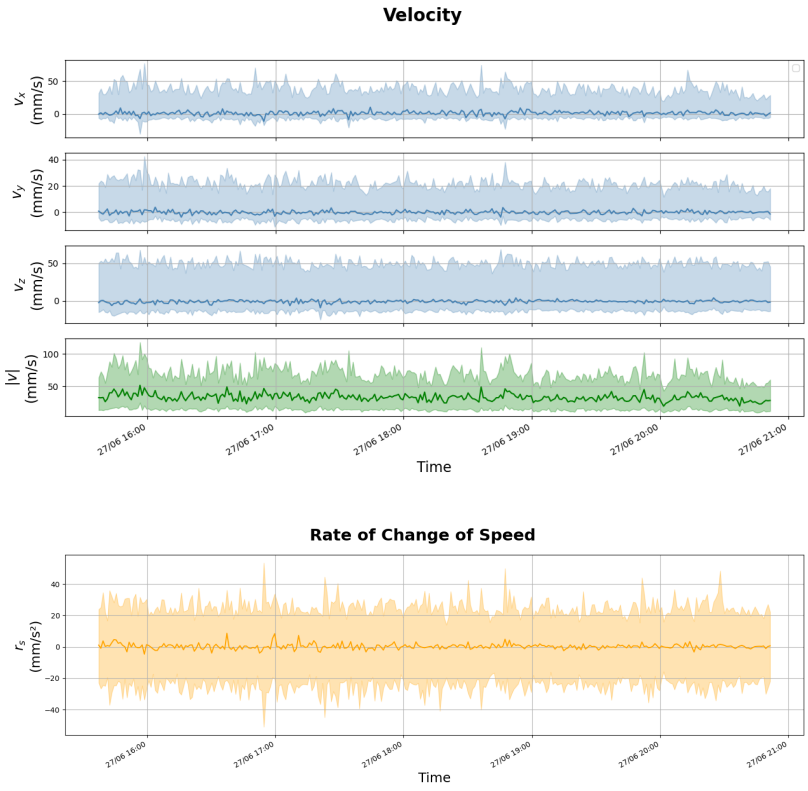


Figure 6.1: *27.06.2022*: Velocity and speed change rate data acquired from object tracking.

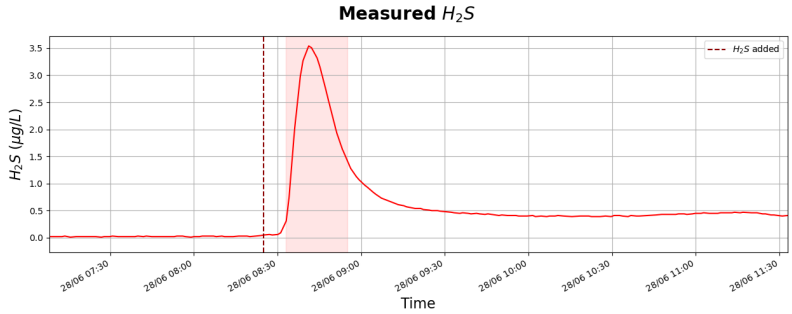


Figure 6.2: *28.06.2022*: Measured H_2S.

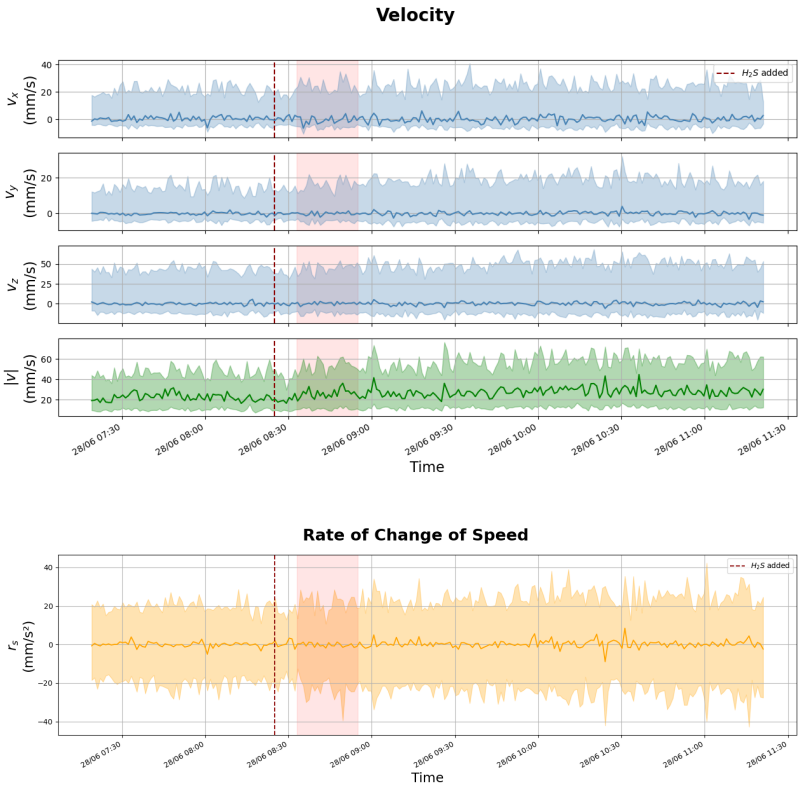


Figure 6.3: *28.06.2022*: Velocity and speed change rate data acquired from object tracking.

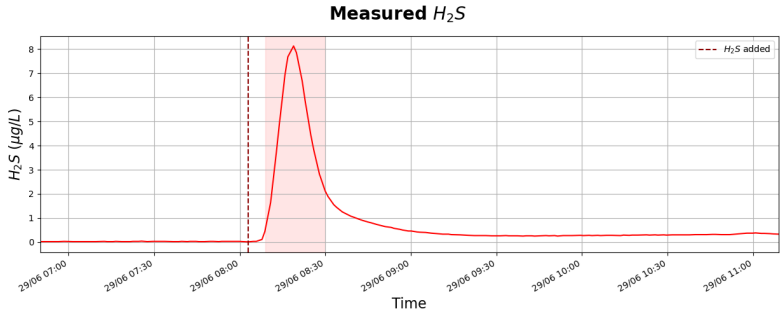


Figure 6.4: *29.06.2022*: Measured H_2S.

Figure 6.5: *29.06.2022*: Velocity and speed change rate data acquired from object tracking.

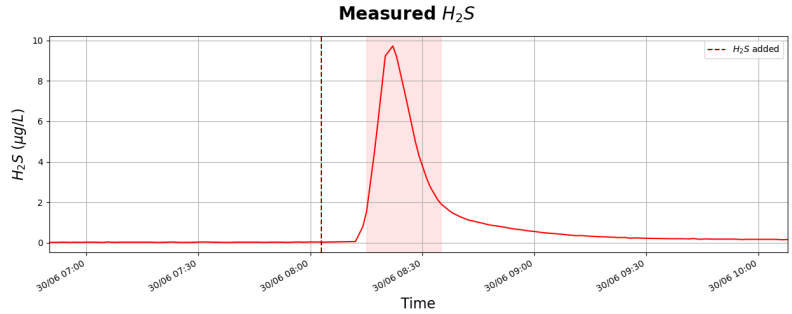


Figure 6.6: *30.06.2022*: Measured H_2S.

Figure 6.7: *30.06.2022*: Velocity and speed change rate data acquired from object tracking.

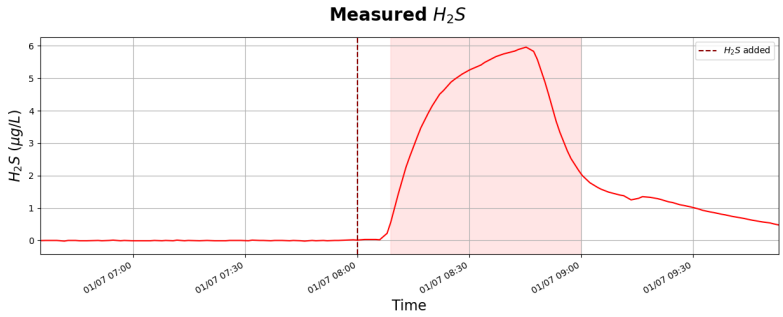


Figure 6.8: *01.07.2022*: Measured H_2S.

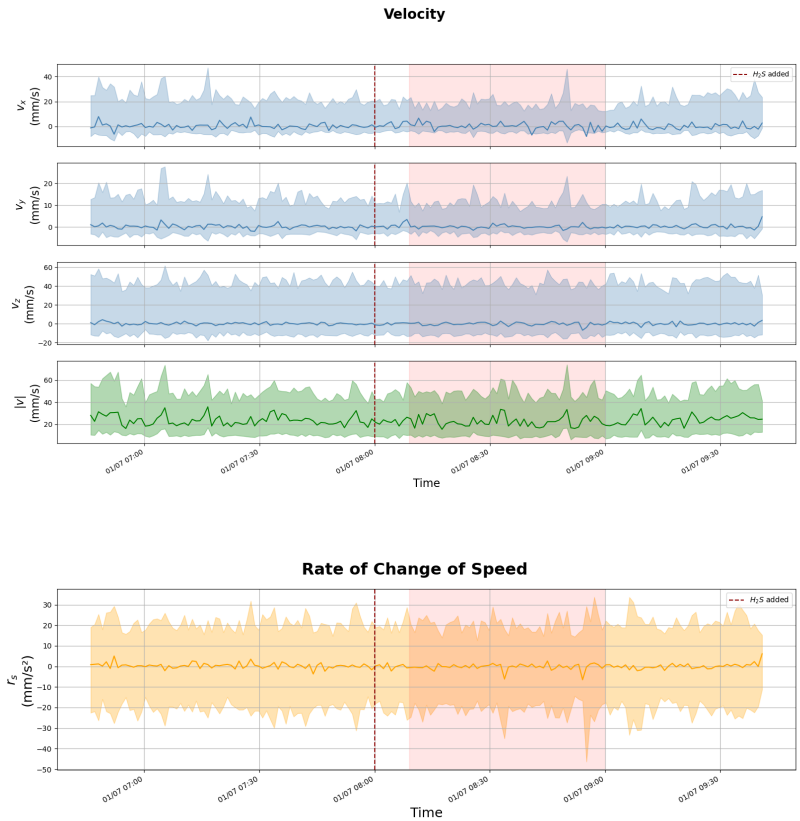


Figure 6.9: *01.07.2022*: Velocity and speed change rate data acquired from object tracking.

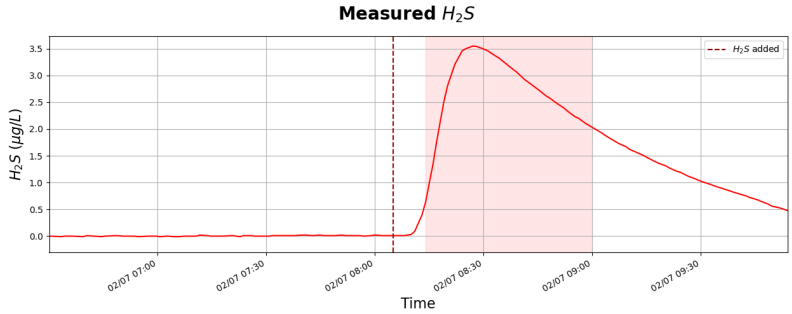


Figure 6.10: *02.07.2022*: Measured H_2S.

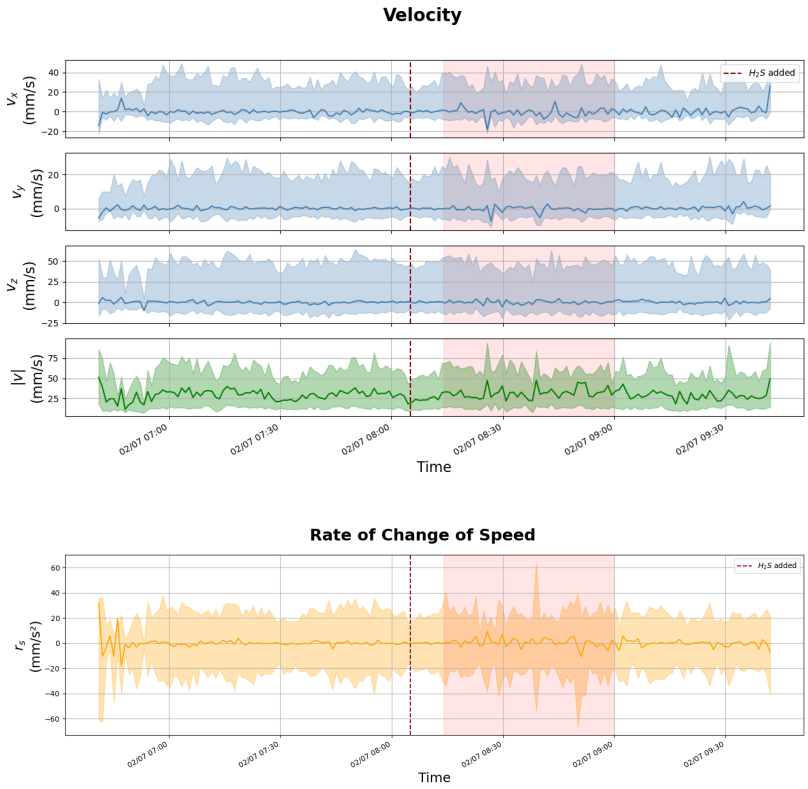


Figure 6.11: *02.07.2022*: Velocity and speed change rate data acquired from object tracking.

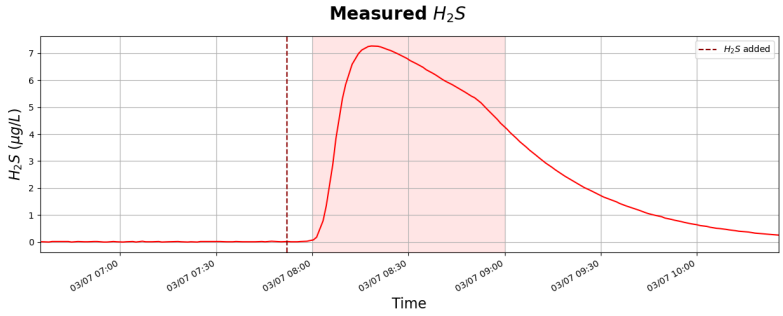


Figure 6.12: *03.07.2022*: Measured H_2S.

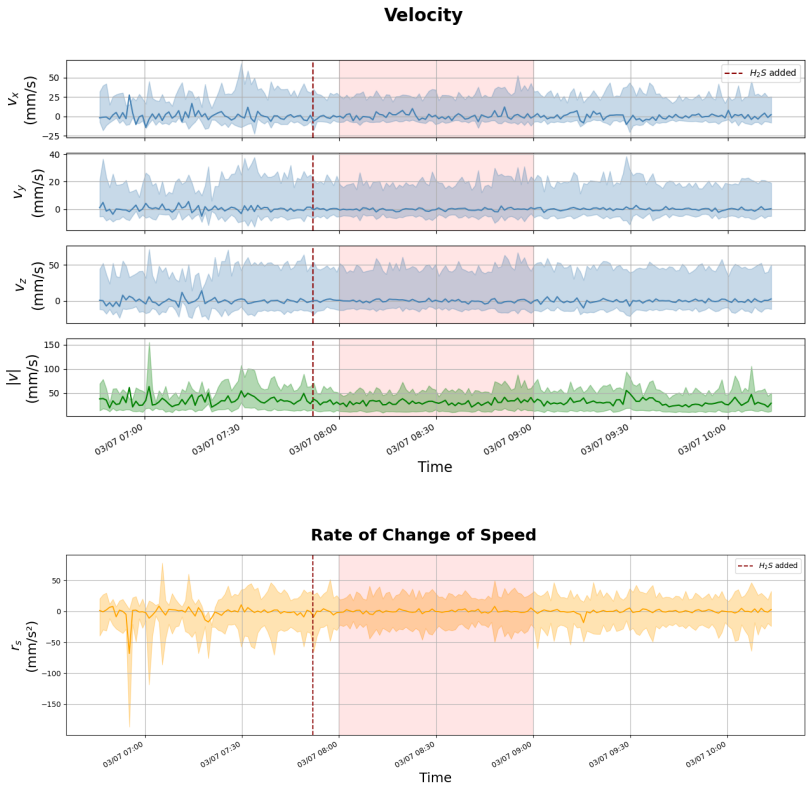


Figure 6.13: *03.07.2022*: Velocity and speed change rate data acquired from object tracking.

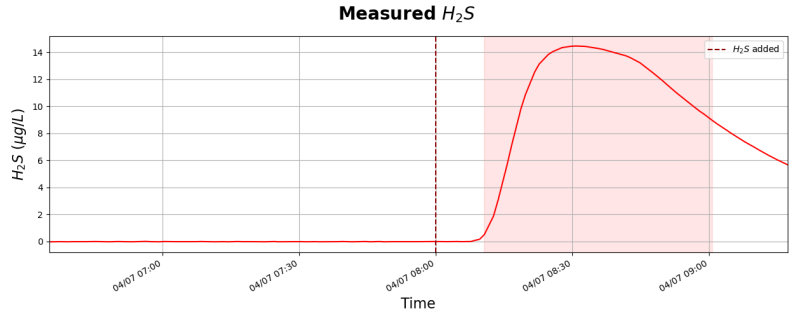


Figure 6.14: *04.07.2022*: Measured H_2S.

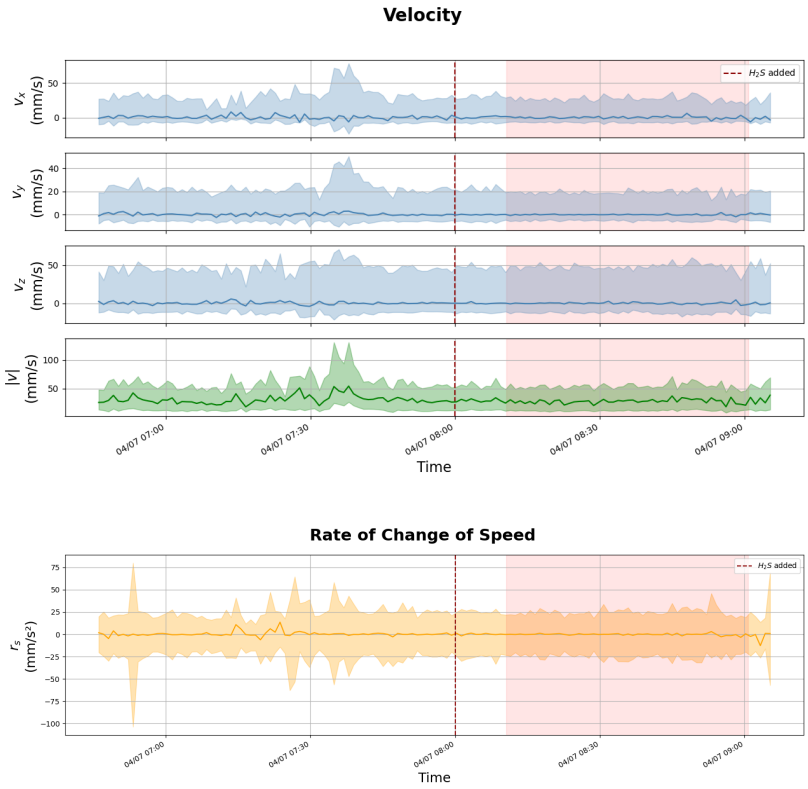


Figure 6.15: *04.07.2022*: Velocity and speed change rate data acquired from object tracking.

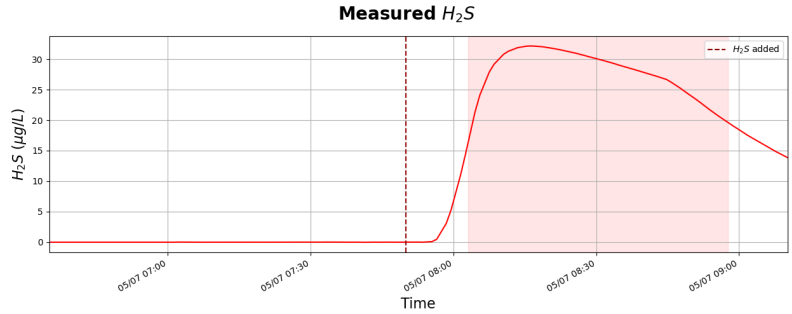


Figure 6.16: *05.07.2022*: Measured H_2S.

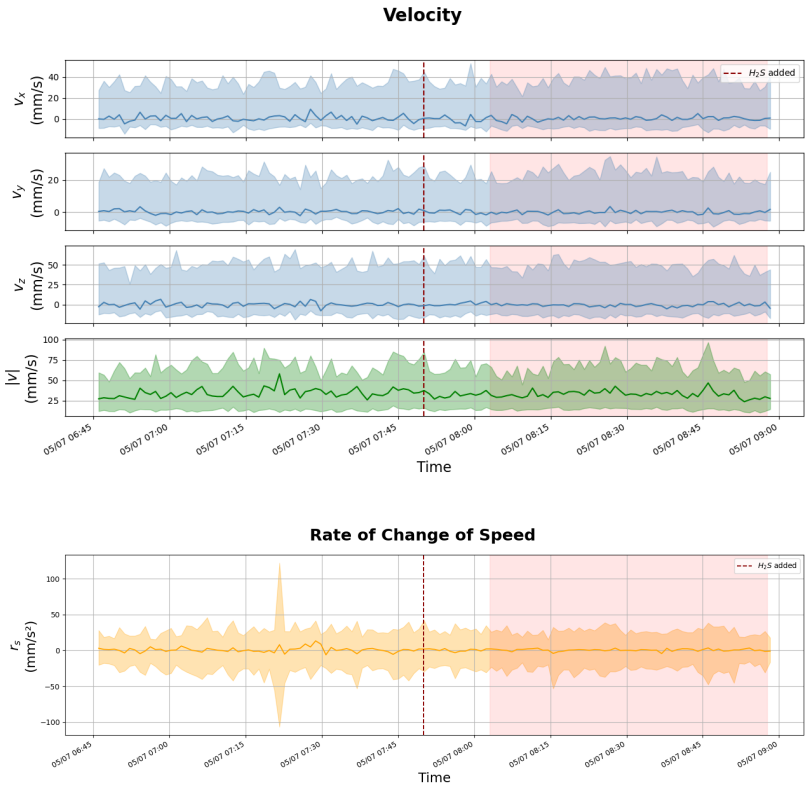


Figure 6.17: *05.07.2022*: Velocity and speed change rate data acquired from object tracking.

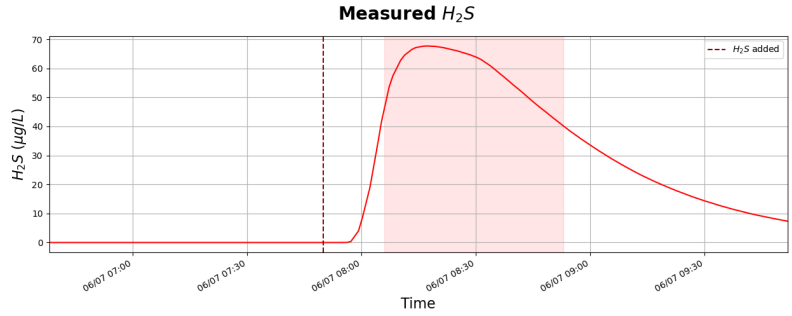


Figure 6.18: *06.07.2022*: Measured H_2S.

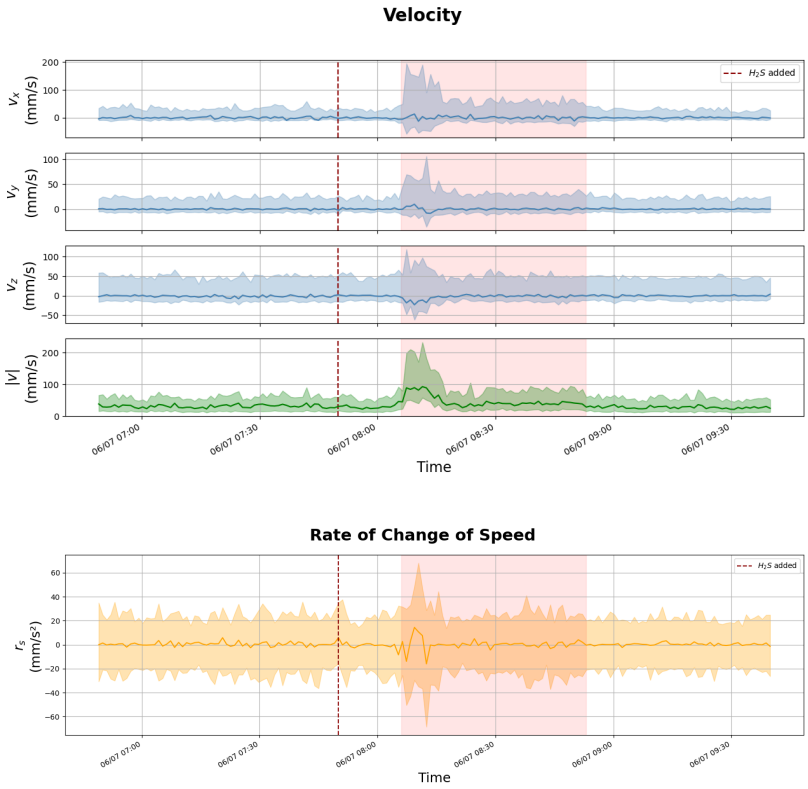


Figure 6.19: *06.07.2022*: Velocity and speed change rate data acquired from object tracking.

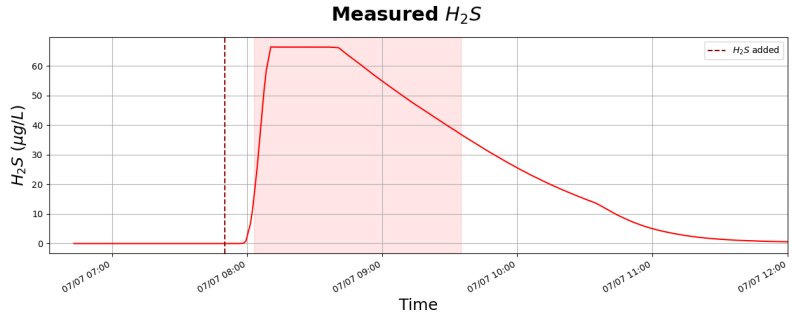


Figure 6.20: *07.07.2022*: Measured H_2S.

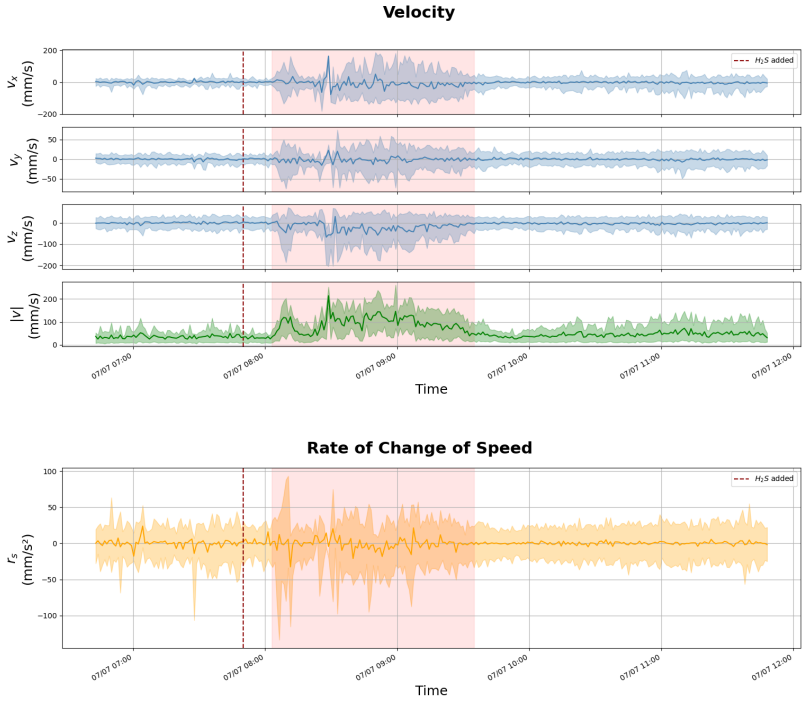


Figure 6.21: *07.07.2022*: Velocity and speed change rate data acquired from object tracking.

B Distribution Data of Fish

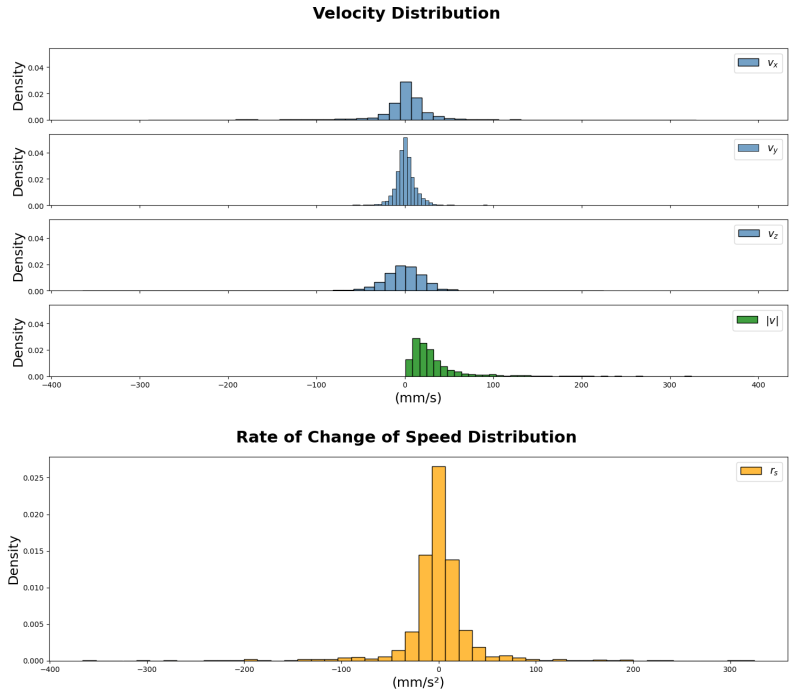


Figure 6.22: Example of a data sample consisting of velocity and speed change rate distributions from *27.06.2022* labeled with $0\mu g/L$. $b_n = 50$.

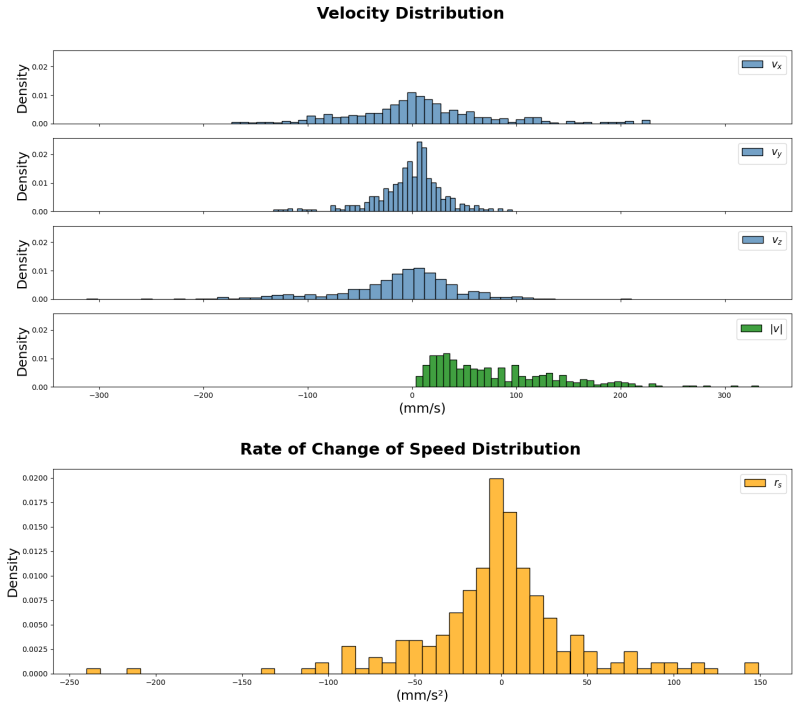


Figure 6.23: Example of a data sample consisting of velocity and speed change rate distributions from *07.07.2022* labeled with $66.4\mu g/L$. $b_n = 50$.

C Estimated 3D Coordinates of Detected Objects

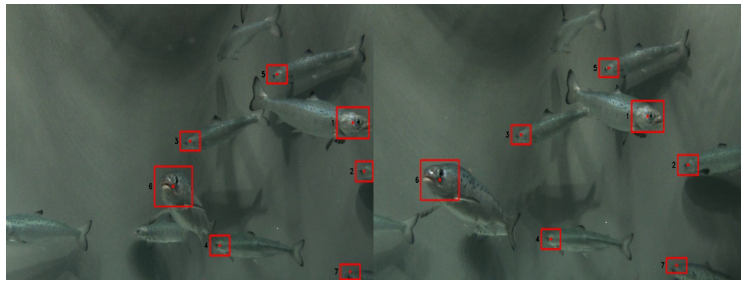


Figure 6.24: *Test image 4*: Detected salmon juveniles in stereo image.

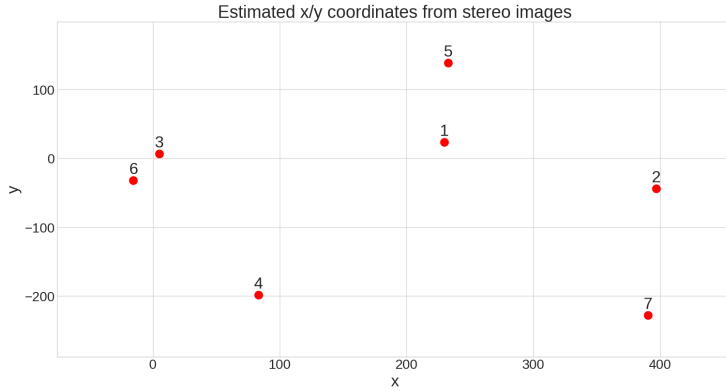


Figure 6.25: *Test image 4*: Estimated x- and y-position of salmon juveniles in stereo image.

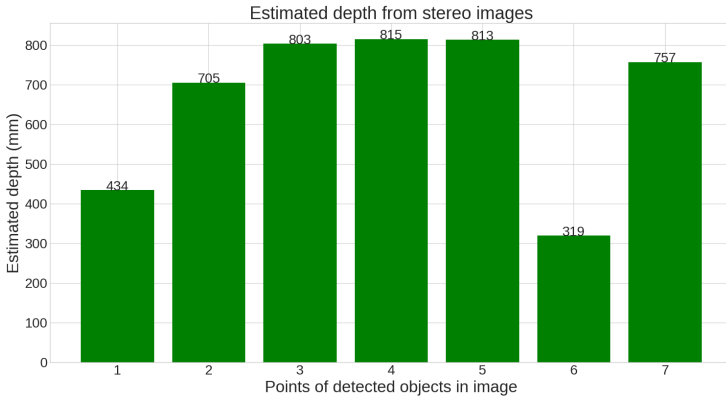


Figure 6.26: *Test image 4*: Estimated depth of salmon juveniles in stereo image.

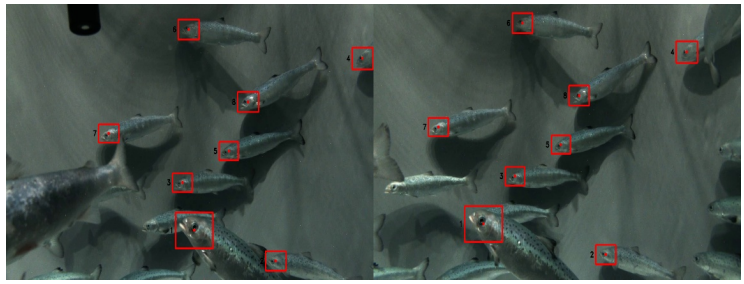


Figure 6.27: *Test image 5*: Detected salmon juveniles in stereo image.

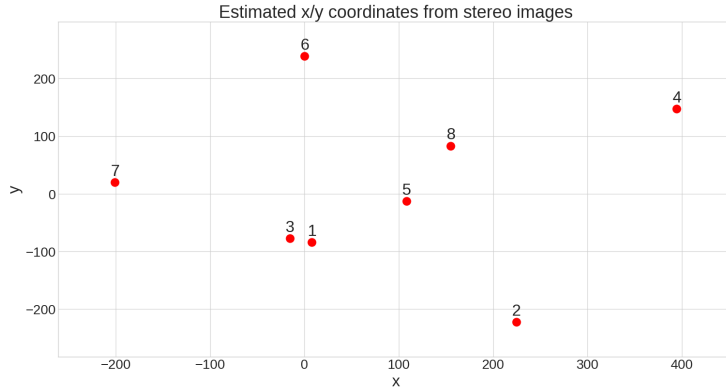


Figure 6.28: *Test image 5*: Estimated x- and y-position of salmon juveniles in stereo image.

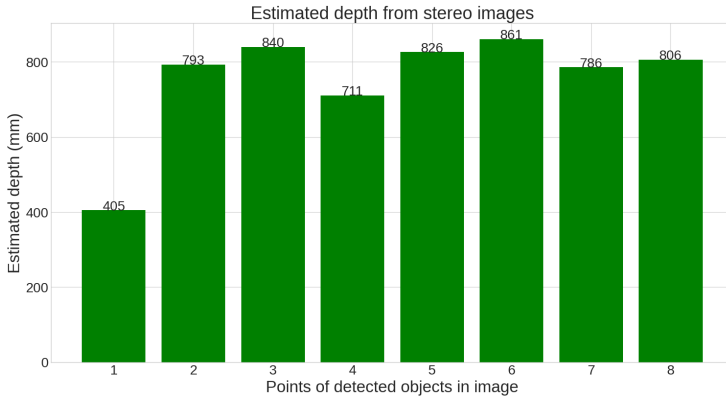


Figure 6.29: *Test image 5*: Estimated depth of salmon juveniles in stereo image.

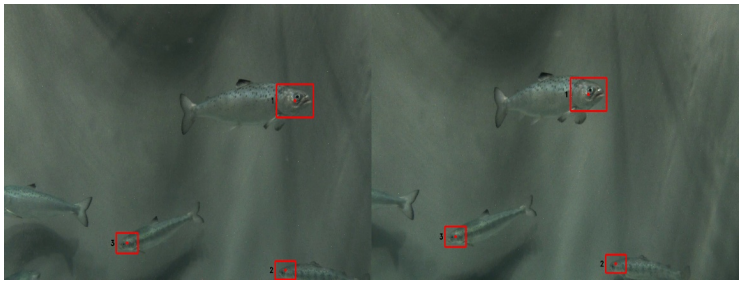


Figure 6.30: *Test image 6*: Detected salmon juveniles in stereo image.

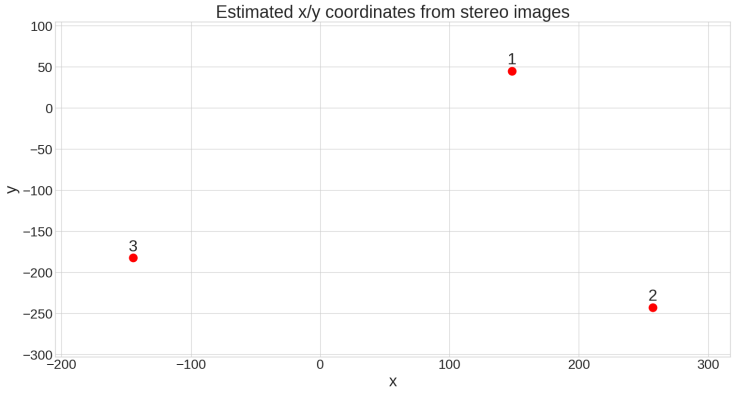


Figure 6.31: *Test image 6*: Estimated x- and y-position of salmon juveniles in stereo image.

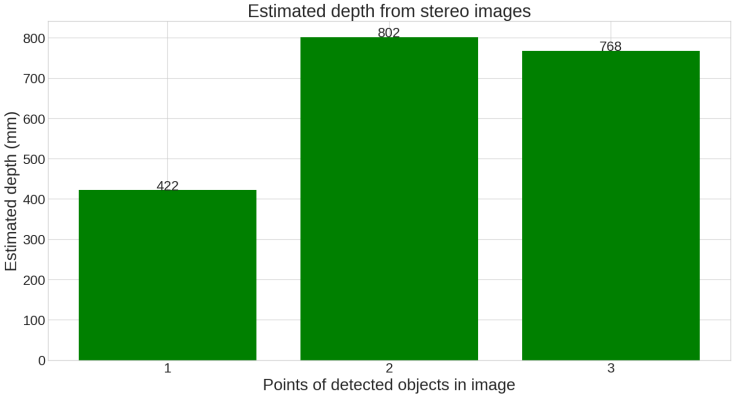


Figure 6.32: *Test image 6*: Estimated depth of salmon juveniles in stereo image.

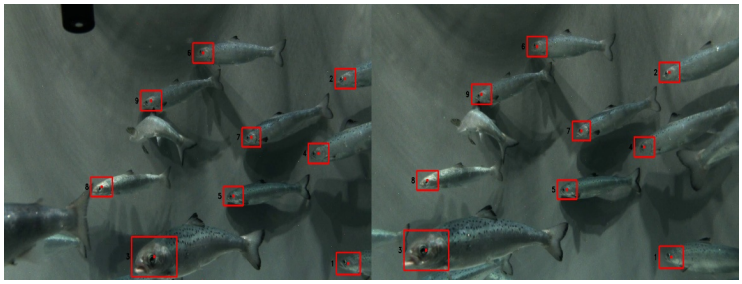


Figure 6.33: *Test image 7*: Detected salmon juveniles in stereo image.

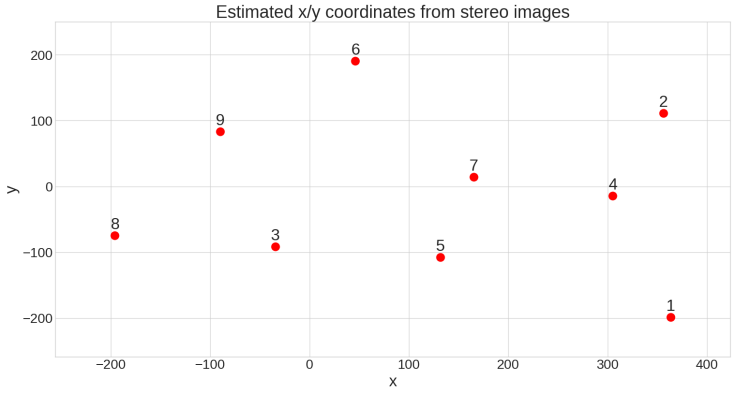


Figure 6.34: *Test image 7*: Estimated x- and y-position of salmon juveniles in stereo image.

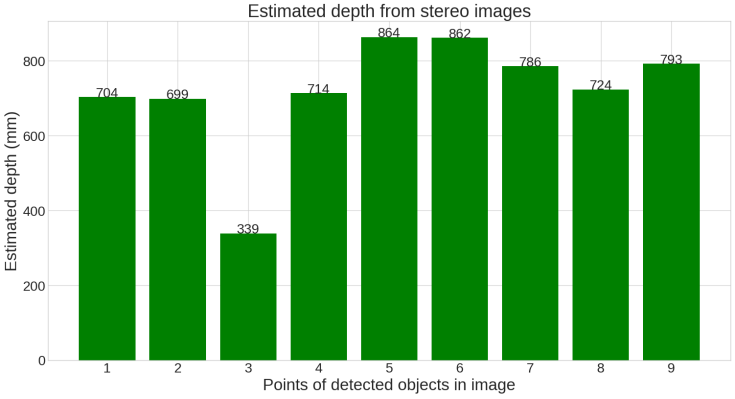


Figure 6.35: *Test image 7*: Estimated depth of salmon juveniles in stereo image.

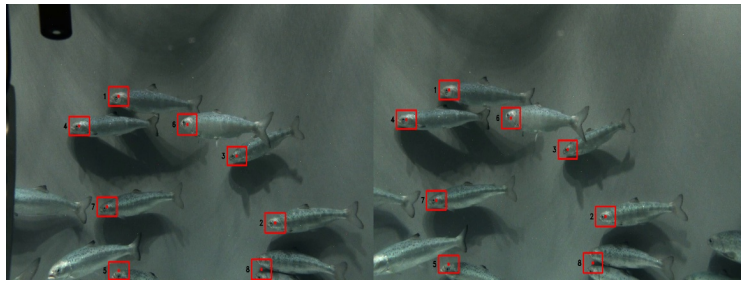


Figure 6.36: *Test image 8*: Detected salmon juveniles in stereo image.

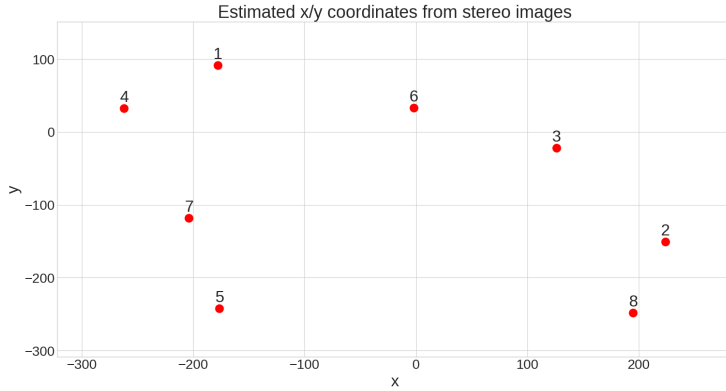


Figure 6.37: *Test image 8*: Estimated x- and y-position of salmon juveniles in stereo image.

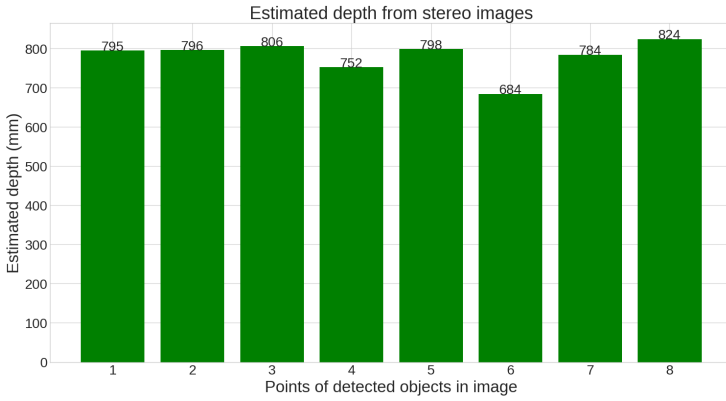


Figure 6.38: *Test image 8*: Estimated depth of salmon juveniles in stereo image.

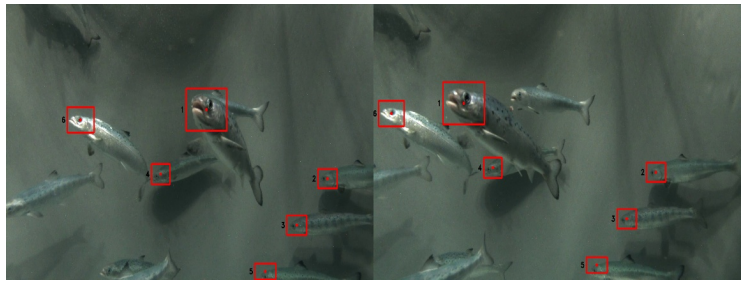


Figure 6.39: *Test image 9*: Detected salmon juveniles in stereo image.

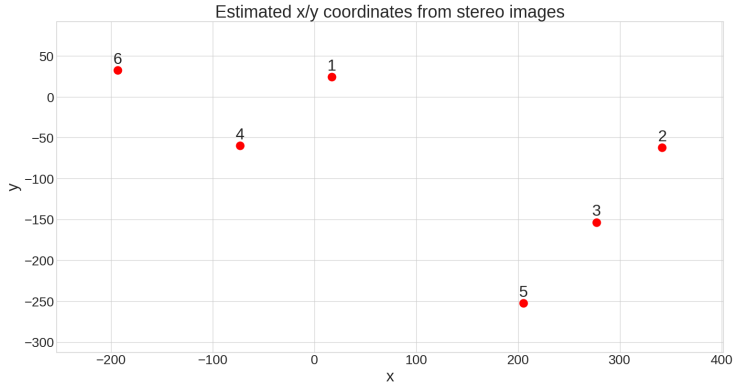


Figure 6.40: *Test image 9*: Estimated x- and y-position of salmon juveniles in stereo image.

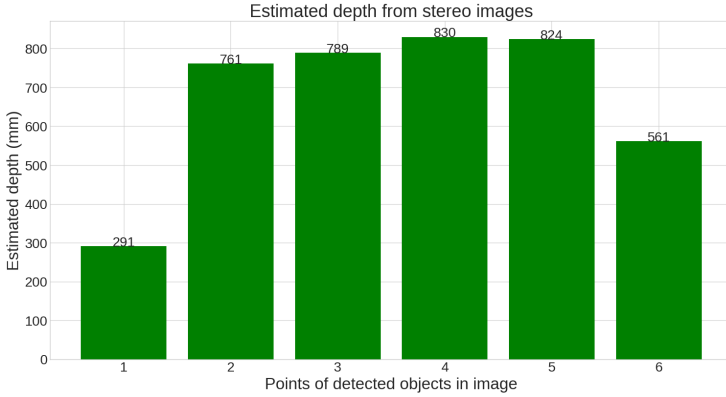


Figure 6.41: *Test image 9*: Estimated depth of salmon juveniles in stereo image.

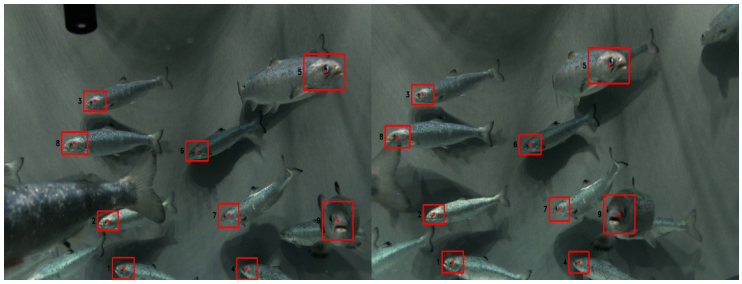


Figure 6.42: *Test image 10*: Detected salmon juveniles in stereo image.

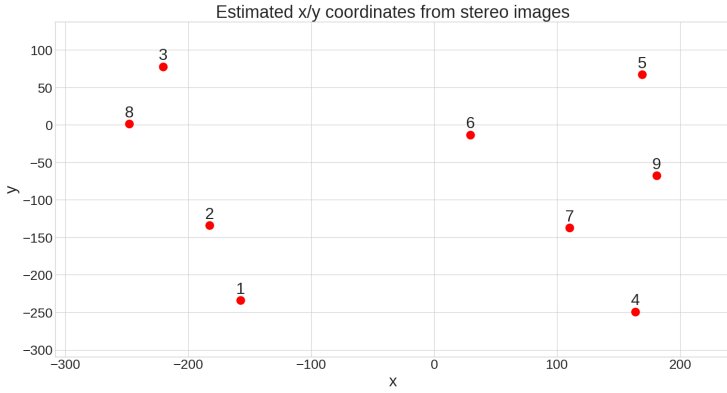


Figure 6.43: *Test image 10*: Estimated x- and y-position of salmon juveniles in stereo image.

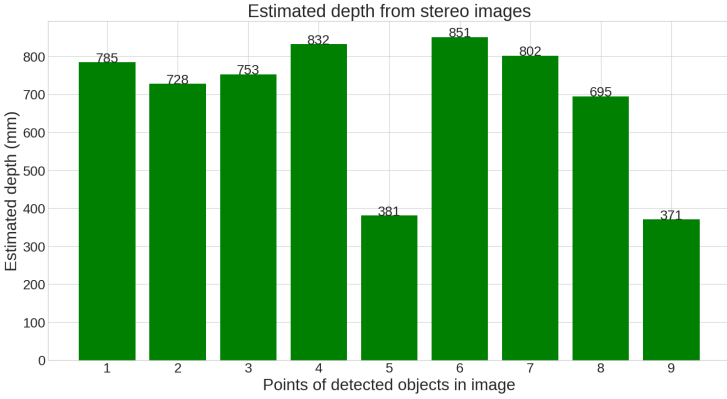


Figure 6.44: *Test image 10*: Estimated depth of salmon juveniles in stereo image.

D Time Series of Positional Data

Variation of T_{lost} Threshold

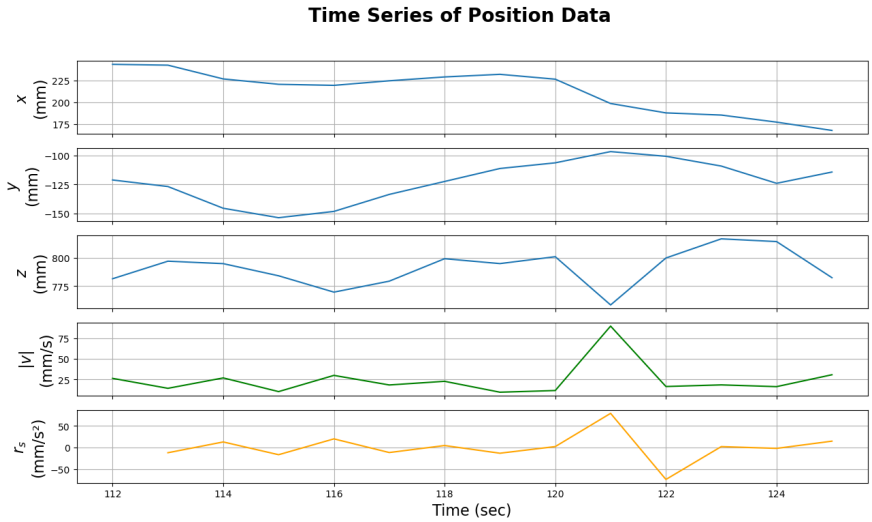


Figure 6.45: $T_{lost} = 1$, $f_c = 15$ frames: Example of time series of positional data of a fish.

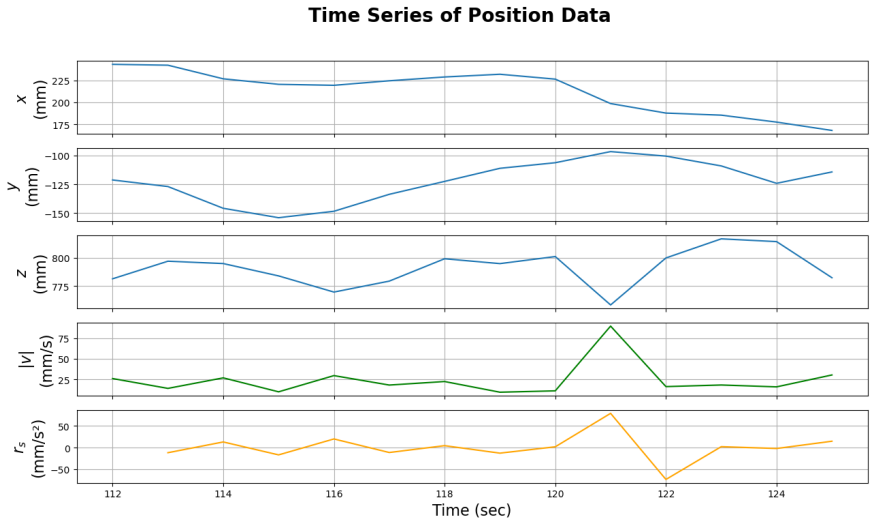


Figure 6.46: $T_{lost} = 5$, $f_c = 15$ frames: Example of time series of positional data of a fish.

Figure 6.47: $T_{lost} = 20$, $f_c = 15$ frames: Example of time series of positional data of a fish.

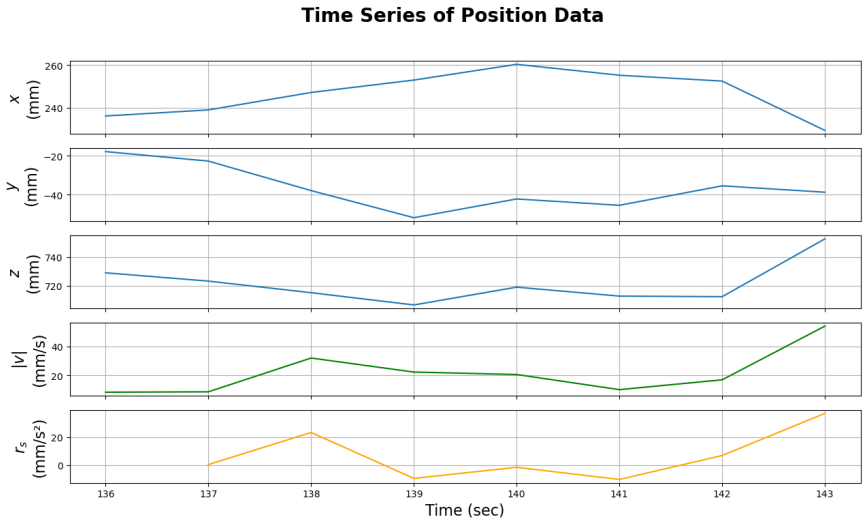


Figure 6.48: $T_{lost} = 1$, $f_c = 15$ frames: Example of time series of positional data of a fish.

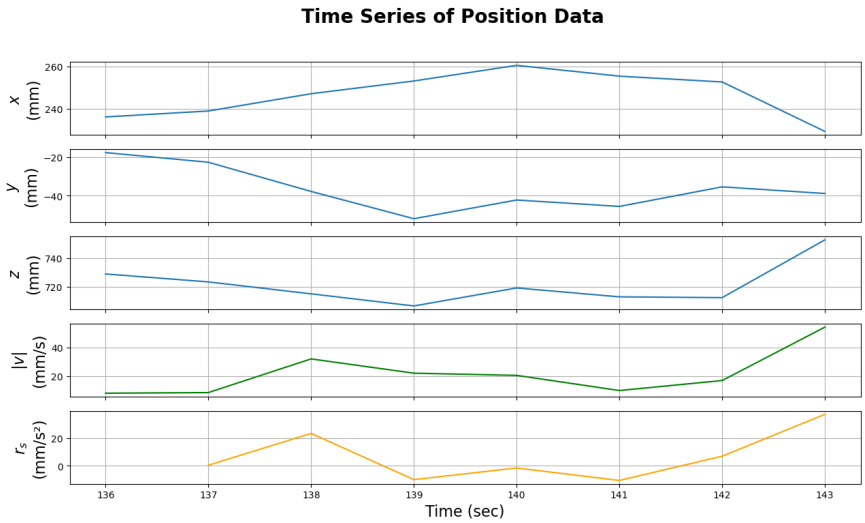


Figure 6.49: $T_{lost} = 5$, $f_c = 15$ frames: Example of time series of positional data of a fish.

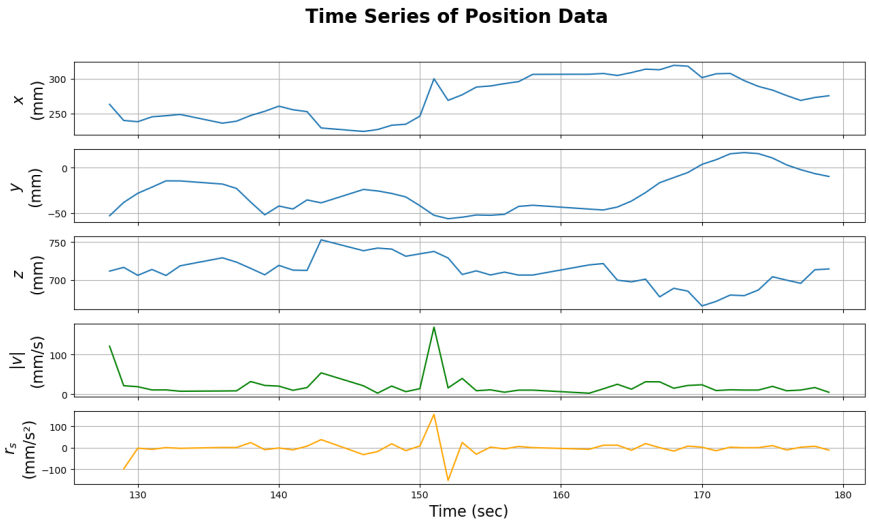


Figure 6.50: $T_{lost} = 20$, $f_c = 15$ frames: Example of time series of positional data of a fish.

Variation of Frame Count

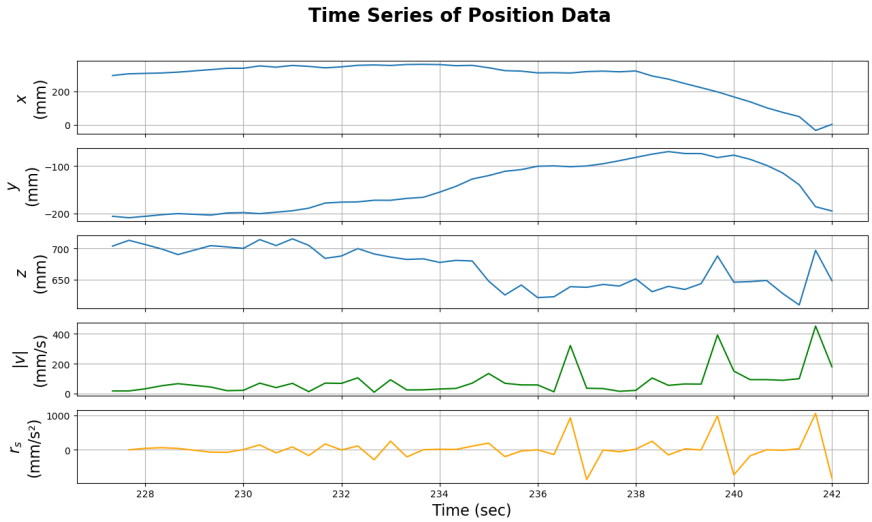


Figure 6.51: $T_{lost} = 5$, $f_c = 5$ frames: Example of time series of positional data of a fish.

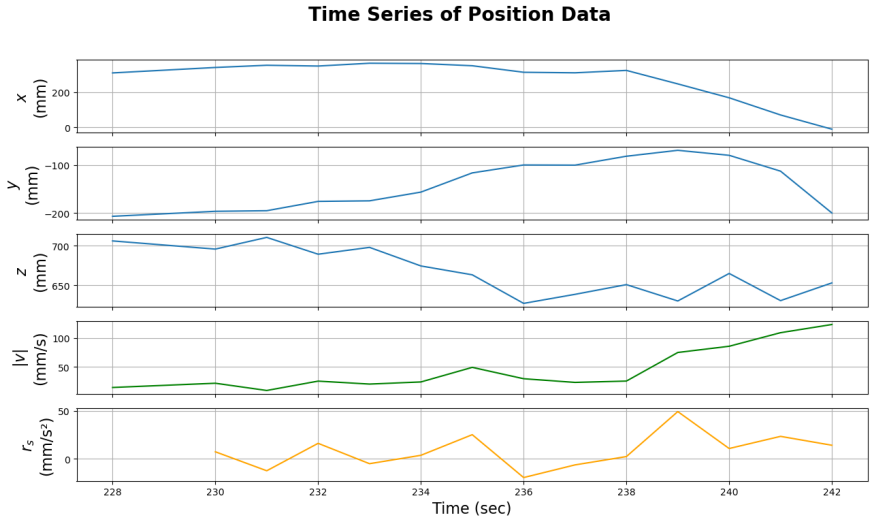


Figure 6.52: $T_{lost} = 5$, $f_c = 15$ frames: Example of time series of positional data of a fish.

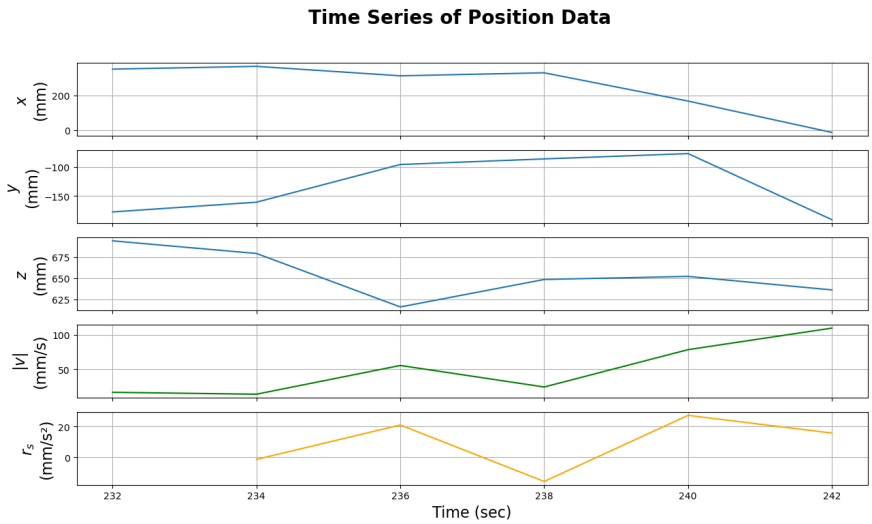


Figure 6.53: $T_{lost} = 5$, $f_c = 30$ frames: Example of time series of positional data of a fish.

Figure 6.54: $T_{lost} = 5$, $f_c = 5$ frames: Example of time series of positional data of a fish.

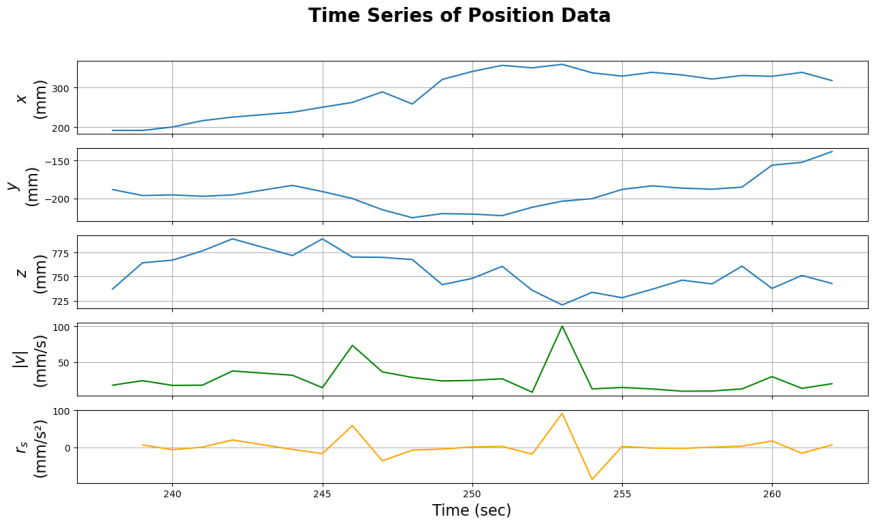


Figure 6.55: $T_{lost} = 5$, $f_c = 15$ frames: Example of time series of positional data of a fish.

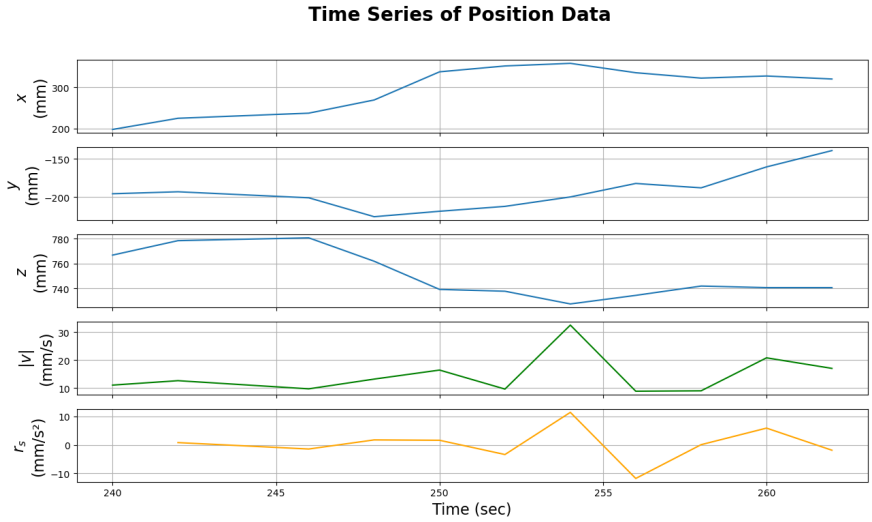
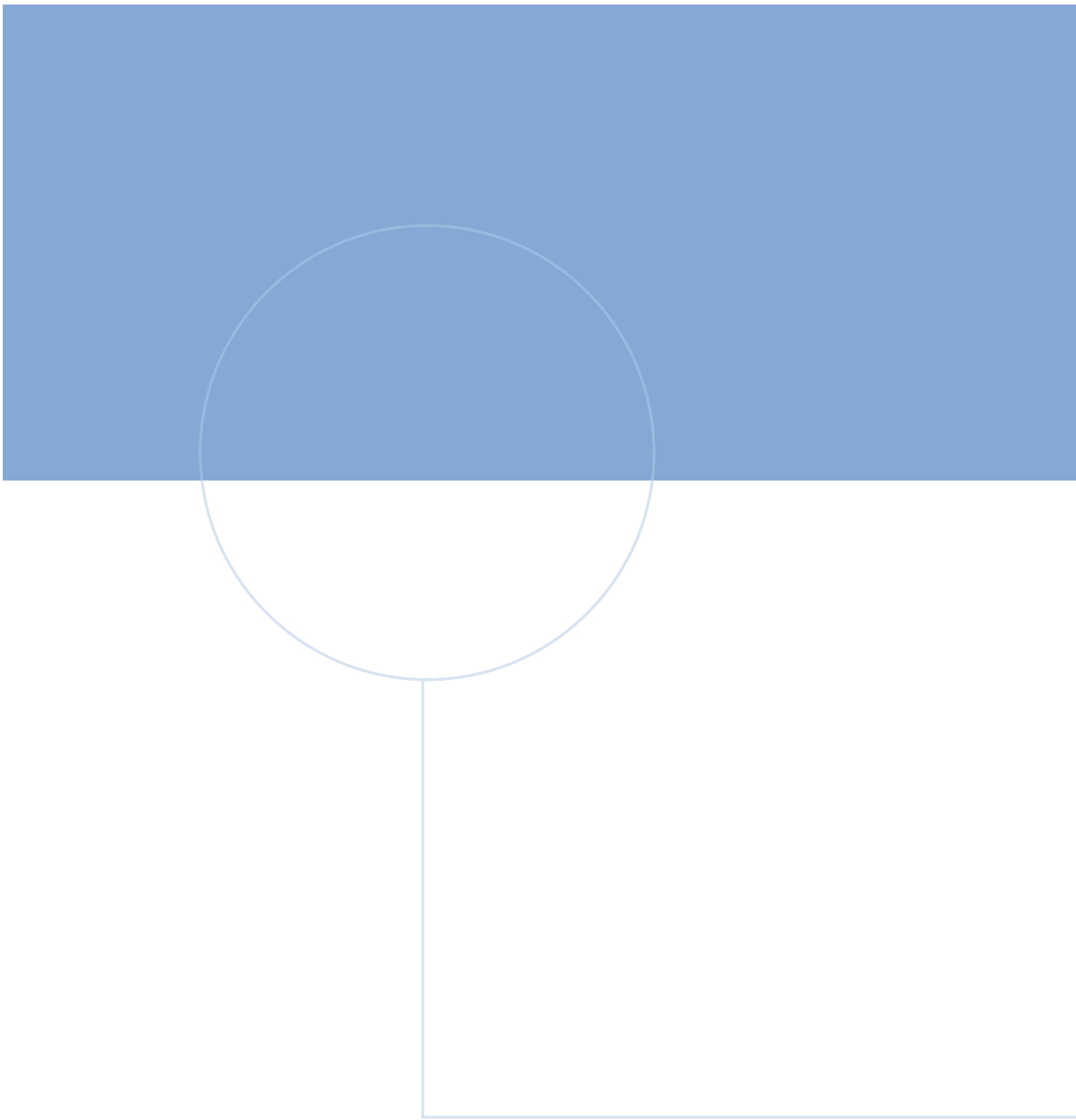


Figure 6.56: $T_{lost} = 5$, $f_c = 30$ frames: Example of time series of positional data of a fish.



NTNU
Norwegian University of
Science and Technology

SINTEF