TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Sanan Suleymanov 213860IASM

UNDERWATER VIDEO OBJECT TRACKING IN PRESENCE OF CHALLENGING VISIBILITY CONDITIONS

Master's Thesis

Supervisor: Elizaveta Dubrovinskaya
Ph.D.

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Sanan Suleymanov 213860IASM

VEEALUSTE OBJEKTIDE VIDEO JÄLGIMINE KEERULISTE NÄHTAVUSTINGIMUSTE KORRAL

Magistritöö

Juhendaja: Elizaveta Dubrovinskaya
Ph.D.

Tallinn 2023

Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Sanan Suleymanov

08.05.2023

Abstract

The biodiversity of marine habitats is threatened by climate change and human activities. To address this problem, it is essential to understand the factors that affect and mitigate the negative impacts. One of the effective ways to understand the changes in biodiversity is to analyze the behavior of animals. For instance, fish migration and behavior can reveal the consequences of climate change and habitat degradation.

This thesis work aims to use computer vision for solving the problem of multiple object tracking in underwater conditions. Tracking underwater objects, especially fish, poses many challenges due to the complex and dynamic nature of underwater environment, such as visual similarity of underwater objects and often insufficient natural lighting. This study examines these challenges and explores how tracking algorithms can be applied in underwater conditions. Four algorithms are compared: DeepSORT, StrongSORT, OCSort, and ByteTRACK. The comparison is based on different metrics that measure the performance of the algorithms on a custom dataset. The results show that OCSort outperforms the other algorithms in terms of accuracy and efficiency. Moreover, the study also presents a web application and an API that demonstrate the use of the chosen tracking technique and make the solution more accessible.

The main contribution of this thesis work is the application of state-of-the-art tracking algorithms to the problem of underwater multiple object tracking, with the goal of replacing the tedious manual tasks done by humans for analysing underwater habitats. The developed software enables users to obtain the tracking results by simply uploading the recorded underwater video, which is a valuable advantage for efficient analysis of biodiversity.

The thesis is written in English and is 42 pages long, including 6 chapters, 13 figures and 4 tables.

Annotatsioon
Veealuste objektide video jälgimine keeruliste nähtavustingimuste korral

Mereelupaikade bioloogilist mitmekesisust ohustavad kliimamuutused ja inimtegevus. Selle probleemiga tegelemiseks on oluline mõista tegureid, mis mõjutavad ja leevendavad negatiivset mõju. Üks tõhus viis bioloogilise mitmekesisuse muutuste mõistmiseks on analüüsida loomade käitumist. Näiteks kalade ränne ja käitumine võib paljastada kliimamuutuste ja elupaikade halvenemise tagajärgi. Käesoleva lõputöö eesmärk on kasutada arvutinägemist mitme objekti jälgimise probleemi lahendamiseks veealustes tingimustes. Veealuste objektide, eriti kalade jälgimine tekitab palju probleeme, mis tulenevad veealuse keskkonna keerukast ja dünaamilisest olemusest, näiteks veealuste objektide visuaalsest sarnasusest ja sageli ebapiisavast looduslikust valgustusest. Käesolevas uuringus uuritakse neid probleeme ja uuritakse, kuidas jälgimisalgoritme saab kohaldada veealustes tingimustes. Võrreldakse nelja algoritmi: DeepSORT, StrongSORT, OCSort ja ByteTRACK. Võrdlus põhineb erinevatel mõõdikutel, mis mõõdavad algoritmide jõudlust kohandatud andmestikul. Tulemused näitavad, et OCSort on teistest algoritmidest täpsuse ja tõhususe poolest parem. Lisaks esitatakse uuringus ka veebirakendus ja API, mis demonstreerivad valitud jälgimistehnika kasutamist ja muudavad lahenduse kättesaadavamaks. Käesoleva lõputöö peamine panus on moodsaimate jälgimisalgoritmide rakendamine veealuse mitme objekti jälgimise probleemile, mille eesmärk on asendada inimeste poolt veealuste elupaikade analüüsimisel tehtavaid tüütuid käsitsi tehtavaid ülesandeid. Välja töötatud tarkvara võimaldab kasutajatel saada jälgimise tulemusi lihtsalt salvestatud veealuse video üleslaadimisega, mis on väärtuslik eelis bioloogilise mitmekesisuse tõhusaks analüüsiks.

Lõputöö on kirjutatud ingilise keeles ning sisaldab teksti 42 leheküljel, 6 peatükki, 13 joonist, 4 tabelit.

List of Abbreviations and Terms

AFLink	Appearance-free link
API	Application Programming Interface
BfG	The German Federal Institute of Hydrology
CNN	Convolutional Neural Network
CVAT	Computer Vision Annotation Tool
ECC	Enhanced Correlation Coefficient
GMM	Gaussian Mixture Model
IDsw	Identity Switch
IDTP	Identification True Positives
IoU	Intersection over Union
JSON	Javascipt Object Notation
KF	Kalman filter
KCF	Kernelized Correlation Filter
LSTM	Long Short-term Memory
MAP	Maximum a Posteriori
mAP	Minimum Average Precision
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
RoI	Region of Interest
OCM	Observation-Centric Momentum
ORU	Observation-Centric Re-Update
reID	Re-Identification
RGB	Red Green Blue
SOT	Single Object Tracking
SOTA	state-of-the-art
MOT	Multiple Object Tracking
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

1. Introduction

The diversity of marine life is essential for the nutritional, economic, recreational, and health necessities of billions of people. However, climate change and also, human-related activities alter the biodiversity of marine habitats according to the evidence. To understand this change and also, predict the source of the changes, it is required to monitor the distribution, abundance, diversity, and health of the organisms [1, 2]. Among common methods such as fish tagging and catch-and-release fishing [3], video-based monitoring can provide timely information about the tracking of the population changes of fish by eliminating manual tasks.

This thesis work discusses in detail the application of computer vision techniques to solve the problems of underwater object tracking as low visibility conditions, insufficient illumination, and turbid water. Due to these environmental conditions of the underwater environment, there are originated challenges such as image lighting and clarity which affect the performance of the proposed methods. Moreover, fish tracking has its difficulties. Because of the acceleration, occlusion and also, and visual similarities of fish, it is more challenging to track rather than general domain objects [4].

Object tracking is one of the well-known and challenging task in computer vision. Object tracking aims to estimate the states of a target object in subsequent frames of a video, based on its initialized state (such as position and size) in a given frame [5]. There are different visual object tracking types, which are single object tracking (SOT) [6], multiple object tracking (MOT) [7], video object segmentation [8] and 3D object tracking [9, 6]. In this work, the MOT task is selected to solve the underwater fish tracking problem. Multi-target tracking aims to give the trajectories of moving objects accurately from given observations. The calculated trajectories are used for the prediction of positions or re-identification. Multiple Object Tracking algorithms can be divided into two categories based on data processing style: the online techniques which process the current video frame in sequence, it is typically used for time-critical applications, and offline techniques that utilize whole video frames [10].

In the study [11], a Kernelized Correlation Filter (KCF) with appearance models of normal and abnormal fish appearance is proposed to deal with the restriction of the KCF model in the sudden body deformation of fish. However, KCF is a single object-tracking algorithm and the dataset used in this study only contains one fish per video.

MOT tracking commonly makes use of the tracking-by-detection approach. To do this, an object detector must be applied to each frame of a video, and then a tracker must be used to connect the items that are found to the tracks that they belonged to [12]. Object detection is a crucial computer vision task that solves the detection of visual object instances of certain classes in images [13]. One of the strongest arguments justifying the implementation of the tracking-by-detection method is given in [14]. The authors used a graph-based two-staged approach for the association of detection results and convolutional neural network (CNN) activations for modeling object appearance. The experimental results were compared to the other trackers which used similar approaches.

The goal of MOT is to track many objects in a video stream at the same time, where the number of targets, their motion patterns, and their appearance can change over time. To do this, modern MOT algorithms are often developed on the base of the tracking-by-detection paradigm [15], which includes the detection of objects and associating them in each frame of the video. Section 2 of this thesis work discusses in detail the online multiple object tracking (MOT) algorithms that are employed to address this difficult challenge. In particular, we examine four MOT algorithms, DeepSORT, StrongSORT, ByteTrack, and OCSort because of their performance in MOT17 and MOT20, and also, DanceTrack benchmark, which provides a platform for the development of MOT algorithms relied on the motion analysis and visual discrimination. The output of the YOLOv7 detection model is used in all of the investigated approaches. The DanceTrack benchmark aims to address challenges that are also present in fish tracking, such as tracking objects with similar appearances and diverse movements. YOLO (You Only Look Once) is a system for real-time object detection which is unlike other models, divides an image into regions and predicts bounding boxes and probabilities for each region using a single neural network [16].

The ultimate goal of this thesis work is to try state-of-the-art (SOTA) MOT methodologies for solving the underwater object tracking problem and it is achieved by carrying out the following steps:

1. Investigation of challenges in underwater fish tracking
2. Conducting literature review on recent advances in object tracking algorithms, knowing evaluation metrics of performance and the format of the dataset construction
3. Preparation of MOT dataset using the underwater fish videos received from BfG [17] for training and evaluation, and the development of data pipeline
4. Application of SOTA MOT algorithms and the comparison of them using performance indicator metrics.
5. Deployment of the selected multiple object tracking algorithms in web application

To make SOTA MOT algorithms accessible for researchers without programming experience or external applications, the web application and API developed are developed in the Flask environment. In this way, the multiple object tracking algorithm is deployed and made it easy to interact with users. The web application allows users to upload underwater videos, analyze them with the multiple object tracking algorithm, and save the tracking results in JSON format. The API extension of the application allows to give the path of the video which runs the tracking algorithm in the server and the result is published in JSON format. A detailed description of the application is mentioned in Section 3.

Most of the works done [11, 14, 18, 19] for solving the underwater object tracking task, tried to develop algorithms that can work efficiently specific to this domain. However, some of these methods focus on the particular challenge in this problem and, has lack attributes to cover the whole multiple object tracking task which is mentioned with all aspects in Section 2. This thesis work aims to utilize computer vision for the tracking of objects in underwater conditions by experimenting with SOTA MOT algorithms to encompass all points of the MOT challenge in this task.

2. Literature review

This section reviews the literature on the object tracking methods used to solve the underwater object tracking problem. The structure of the literature review is as follows. The general overview of the issues with underwater object tracking is provided in Section 2.1. The approaches used to address this difficulty are comprehensively detailed in Section 2.2. Section 2.3 described the detection models used in underwater object tracking.

2.1 Key problems in underwater fish tracking

A rising number of researchers utilize video-based underwater monitoring in order to track changes in the underwater ecosystem over time. The health of ecosystem can be determined in a way by analyzing underwater fauna and flora, particularly fish. In fact, large-scale video collection of the fish populations is possible, but manual analysis by human experts is time and also, money consuming [14].

To track fish movements and gauge any uncertainty related to those movements, marine biologists typically use electronic tags such as passive integrated transponder tags, visible implant fluorescent elastomer tags, and acoustic tags. These techniques involve surgically implanting, injecting, feeding, or externally attaching tags to fish, making them intrusive. This takes a lot of time and only gathers a little amount of data. Tagging must be done under skilled supervision, which is not always possible. This problem drives the need for non-intrusive fish tracking techniques. Due to recent advancements in the technology of underwater cameras, tracking that incorporates vision-based measurement (VBM) is a very promising technique [18].

Computer-aided tracking of numerous identical-sized zebrafish with maintained identification after occlusion is an open issue in ethology. When tracking a huge number of objects, cutting-edge systems cannot maintain correct IDs over an extended period of time following severe occlusion. For the purpose of analyzing individual behavior, it is essential to track numerous zebrafish while correctly maintaining identity throughout the entire movie. A situation of occlusion will occur in the top camera's field of view when more than two fish physically engage with or swim over one another. Three scenarios are possible following occlusion. In the first case, a new identification is determined after occlusion rather than the fish's identity being kept. In the second scenario, identities are connected across occlusions using a prediction framework based on an individual movement prediction mechanism, such as a Kalman filter [20] or particle filter. The first scenario occurs if occlusions are too complex to link identities after occlusion. At this point, manual review and correction should be undertaken. In the third case, there are no mistakes or incorrect identity assignments and the trajectory still corresponds to the initial identity prior to occlusion. Artificial marking is a technique to address this issue, however in other circumstances, such as when the population is large or a fish is too little, this approach is not practical [21].

2.2 Appearance model based tracking

Appearance-based tracking techniques are commonly used in computer vision applications to track objects in video streams. These techniques rely on the visual similarities of the objects based on their features in each frame of the video stream. One of the main challenges in appearance-based tracking is to deal with the locomotion of live animals, which can cause significant changes in their appearance. A study [11] proposed a method for tracking fish that can adapt to the various appearance changes caused by non-rigid deformation. The authors noted that while recent deep learning approaches have improved the tracking algorithms, they are also more time-consuming for real-time tracking. Therefore, they proposed an adaptive multi-appearance model that was compared with the Kernelized Correlation filter (KCF). KCF is a tracking algorithm that is designed to distinguish between the target object and the background. It is a classifier that is trained on scaled and rotated patches [22]. The authors built the appearance model for both normal and abnormal appearances of fish. They defined the normal appearance as linear swimming and the abnormal appearance as turning of fish. The experimental results showed that the proposed method outperformed the KCF model [11] by 13.5% on average in accuracy. However, the proposed method had some limitations. Firstly, the frames used to show the results only contained one fish, which limits the applicability of the method to more complex scenarios with multiple objects. Secondly, the tracker sometimes lost the object, indicating that the proposed method might not be robust enough to handle all possible appearance changes. Finally, there might be difficulties in integrating the tracking results with other computer vision methods because the bounding box size did not always match the size of the tracklets.

Concetto et al. [19] developed a tracking algorithm in order to meet the demand in fish tracking that makes use of covariance representation to link many sorts of features, including position, derivatives, color intensities, and others, as well as to characterize the object's appearance and statistical data. The algorithm's accuracy was assessed using hand-labeled ground truth data which consists of 30000 frames from ten different videos, yielding approximately 94% average performance. This performance was estimated using several ratios that give an indication of how well a tracking algorithm performs both globally (e.g., counting objects over a fixed period) and locally (e.g., in identifying object occlusions). To create a covariance matrix for each detected object, the feature vector is created that contains the coordinates of the pixel, its RGB and hue values, as well as the standard deviation and mean of a 5x5 window's histogram with the target pixel at its center. The covariance matrix that models the object is determined using this feature vector and is connected to the detected object. The object is then compared to the other tracked objects using this matrix to determine which one it most closely looks like. Authors used Förstner's distance instead of Euclidean distance because the covariance matrices are not in Euclidean space. The similarity of two covariance matrices is determined using the following formula [19]:

$$\rho(C_i, C_j) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k(C_i, C_j)} \tag{2.1}$$

where d denotes the matrix order and $\{\lambda_k(C_i, C_j)\}$ is the C_i and C_j covariance matrices' eigenvalues which are calculated from

$$\lambda_k C_i x_k - C_j x_k = 0 \quad k = 1...d \tag{2.2}$$

The algorithm also deals with the loss of tracked fish for solving occlusion problems by using the counter (TTL) for each of the tracklets which counts how many numbers of frames the object missed and whenever the counter of the tracker reaches the value which is given by the user, that object is discarded. The tracker outperforms the compared CAMSHIFT tracker and can accurately detect over 90% of objects With a right choice rate of more than 96%. The experimental results of the proposed algorithm together with CAMSHIFT are shown on the same frames in Fig. 1.

The authors proposed a new deep-learning based solution which is called DFTNet that uses a Siamese network to encode appearance similarity, and an attention-based long short-term memory network (LSTM) is utilized to capture motion similarity across consecutive frames. In order to combine spatial similarity cues in the final result, they additionally apply an intersection-over-union matching score. The suggested approach offers joint optimization scores for keeping tracklet data that encodes motion, appearance, and spatial similarity signals [18]. The Siamese network was first time presented by Bromley and LeCun in the early 1990s to handle the signature verification challenge as an image matching

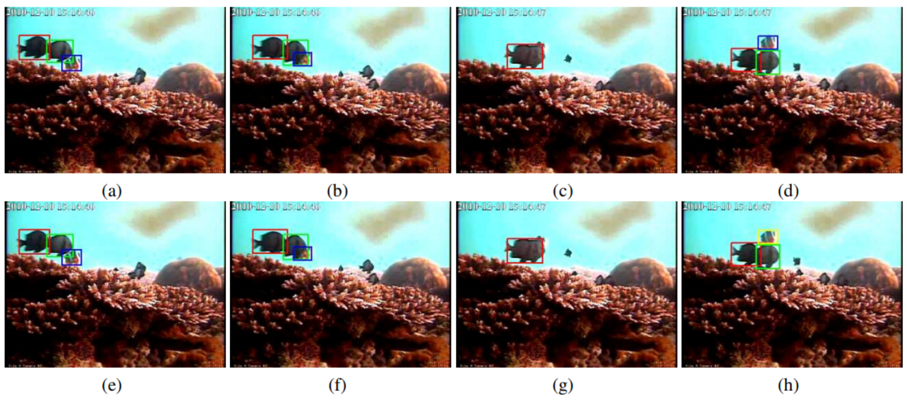


Figure 1. Results of the proposed technique (top frames) and CAMSHIFT (bottom frames) on three occluding fish. It can be observed that the CAMSHIFT tracker is unable to distinguish the fish in the yellow box in Fig.1h which is the same fish in the blue box in Fig.1e [19].

problem [23]. It is basically a neural network made up of two twin networks that each receive different inputs but are connected at the top by an energy function [24]. LSTM is a Recurrent Neural Network proposed by Sepp Hochreiter and Jürgen Schmidhuber which can selectively keep the information over extended periods and is beneficial for tasks utilizing sequential data [25]. Scientists used Fish4knowledge [26] videos for their research and were able to reduce identification switches (IDsw) significantly by 60.9% when compared to other methods [18]. One of the compared methods is the well-known MOT algorithm DeepSORT which is also shown as outperformed by DFTNet. However, the DFTNet is computationally expensive by using different networks for tracking, compare to other MOT algorithms.

The paper [14] discusses the application of two-stage graph approach with the utilization of activations of convolutional neural network (CNN) for modelling object appearances. The algorithm searches for matching detections in several frames that have strong similarities in appearance and position in the first stage and extracts them into small trajectories (Tracklet Extraction). A similar approach is used in the second stage to link the so-called tracklets to complete object trajectories (Tracklet Linking). The algorithm were compared with graph based approaches using MOTA/MOTP and IDsw metrics, which are explained in Section 3.4. The approach produced better results when compared to the work done by Chuang et al.[27] on underwater fish tracking, which employs the k-shortest path optimization method for multiple object tracking [28]. However, it was discovered that the algorithm's performance was so close to the Mothes and Denzler's [29] graph-based technique.

Mygel and Prospero for solving multiple fish tracking challenges developed the synthetic dataset because of the lack of available annotated benchmark [30]. They also present SynDHN, an integrated detector, and tracker that takes advantage of the Deep Hungarian

Network for tracking. The authors demonstrate that SynDHN performs better than baseline techniques and generalizes effectively to real underwater video tracking by taking into account spatial and also, appearance features for affinity estimate. They conduct tracking-by-detection using Faster-RCNN [31] and customized Deep Hungarian Network (DHN). DHN is proposed by Xu et al. [32] and according to the original architecture, it takes a distance matrix D as an input and its row is the predicted bounding box a t time frame and column is the ground truth bounding box at the same time frame. The goal of DHN is to determine the optimal ground truth bounding box match for each predicted bounding box based on the affinities of their features (such as IoU). They repurposed DHN by putting the predicted bounding box at t+1 time frame as a column of matrix D and remaining the column unchanged. The disadvantage of the provided solution is not having a re-identification (reID) feature which can recover the tracks reappeared after the lost in previous frames. The model result is compared by the methods as Tractor [33] and SORT [34] that also, only can perform frame-by-frame tracking, and got higher results in MOT metrics and lower identification switches (IDs).

In the paper [35], authors utilized the centroid tracking method for tracking Roman seabream after detection of them using the Mask R-CNN model. The algorithm uses Euclidean distance measure to compare the centroids of each initial Roman seabream detections in F_t with the centroids of the new detections in F_{t+1}, and it assigns the same ID to the F_{t+1} detections that are closest to the initial F_t detections in pairwise centroid distance. At this point, if the number of detected fishes in F_{t+1} is more than in F_t, new IDs are allocated. The application of the centroid tracking also increases the reliability of the Mask R-CNN model which can be utilized in order to correct false outputs given by Mask R-CNN. However, the performance of the centroid tracking is not evaluated due to the amount of required manual work.

The primary objective of the presented paper[36] is to use simple image processing methods for tracking and counting fish. The conducted research covers obtaining the fish images and processing of the collected images by utilizing blob analysis and the Euclidean filtering techniques. The distance of the object traveled was calculated using the centroid in accordance with the Euclidean distance formula [37]. The variables used were the moving object's pixel coordinates from the beginning to the end. The following is a presentation of the distance calculation algorithm:

1. Check the centroid position of each image.
2. Determine the separation between two centroid images.
3. for X resolution (present_position=initial_value: final_value)
4. for the Y resolution (present_position=initial_value: final_value)

5. Determine the distance change using the formula Distance=p(X2:X1)2 + (Y2:Y1) and Y1 is the prior pixel position and Y2 is the current pixel position in height, and X1 is the previous pixel position and X2 is the present pixel position in width.
6. save each distance value in an array.

Following the filtering procedure, the input image is improved and segmented in accordance with the characteristics of the objects. In order to extract the object of interest, photos are then made grayscale. According to the experimental results, the detection algorithm only seldom experienced overcounting or undercounting problems. It was determined that some detected objects during the procedure were not the objects of interest and this was a false detection that resulted in excessive counting. The shadows cast by the environmental lightning or foreign objects were two examples of overcounting. Furthermore, failure to recognize smaller objects as well as objects that were near to one another also contributed to undercounting[36]. Despite the fact that the results show high accuracy, it is not feasible to apply the given method—which was tested on white-colored fish passes—in natural underwater settings due to the difficult environmental conditions for differentiating fish from one another and for tracking each fish separately due to their intricate mutual movements.

This article [38] focuses on employing stationary and nonstationary camera configurations to address the issue of visual tracking in an underwater environment. Deepak et al. suggests to utilize the YCbCr color representation in order to get scene representation based on the dominant color component. The developed adaptive model chooses Walsh-Hadamard (WH) kernels for the effective extraction of edge, color and also, texture strengths, while a new feature - range strength is presented to extract the intensity variation from underwater frames in the local neighborhood by employing the WH kernel. The particle framework is used for tracking the fish by integrating the probabilities of the feature strengths. Based on the Sϕrensen distance, the reference feature strengths that are utilized to determine the weights for the particles are updated. The target initialization is involved in the first step by manually placing a rectangular blob. The blob's dimension is recorded for use in the future. The assessment of the proposed method is done using the underwater datasets from underwaterchangedetection (UWCD) [39], reefVid [40], National Oceanic and Atmospheric Administration (NOAA) and fish4knowledge (F4K) [26]. According to the conclusion of the authors this method performs effectively under conditions of blur, variable illumination, light haze, partial occlusion, and partial camouflage on the base of the used . In spite of the fact that this technique performs well in the challenging condition of underwater, the model tested in the tracking of single fish and it can have limited capabilities for tracking of multiple fishes because of their complex mutual behaviors. The approach also lacks the reID module for fish re-identification, which can track an object when it returns to the video frame after disappearing for a number of frames.

2.3 Detection as a part of underwater object tracking

In actuality, the tracking-by-detection concept forms the foundation of most state-of-the-art MOT methodologies. The techniques outlined in Section 2.2 demonstrate that this is true for tracking in an underwater object environment as well.

The research [41] involved collecting three different datasets from the real water power plants and training a YOLO deep learning model to detect fishes in underwater videos. The selected model is YOLOv3 which is trained using the transfer learning. With samples from all three datasets, training and testing produced an average precision score of 0.5392. The model was trained with samples from only two datasets and then evaluated with examples from all three to determine how well it generalizes to new datasets. However, the model was unable to recognize fish in the dataset that it had not been trained on. In comparison to the results achieved by the model trained on all three datasets, the mAP scores on the other two datasets utilized in the training set were higher.

YOLO is the conceptualization of object detection problem as a regression to structurally separated bounding boxes and related class probabilities. It uses only a single neural network for predicting bounding boxes and class probabilities on entire images in a single assessment. Since the entire detection pipeline consists of a single network, detection performance may be optimized end-to-end [16].

Another research [42] uses hybrid solution in order to eliminate the underwater challenges such as fish shape deformations while swimming, dynamic backgrounds, murky water, low resolution, fish camouflage, and tiny differences between some fish species. Authors proposed a method to recognize and categorize fishes in underwater videos that combines Gaussian mixture and optical flow models with YOLO network. According to the paper YOLO based object detection is originally used to only detect occurrences of fish that is static and easily observable. YOLO can be able to detect freely swimming fish that are hidden in the background utilizing temporal data collected by optical flow and Gaussian mixture models which overcomes the mentioned limitation. The suggested approach got 95.47% and 91.2% F-scores in fish detection which is evaluated in LifeCLEF 2015 benchmark and dataset received from University of Western Austria.

The work of Xiu Li et al. [43] also contains the application of a deep convolutional network approach in which they used Fast R-CNN for detecting and also, recognizing fish species. The Fast R-CNN model has the convolutional network which produces the feature map from the input image and region of interest (RoI) layer which extracts the feature vector from the feature map for each object proposal received as input. The outputs consist of two layers: one of them is softmax probabilities of detections and another layer that gives the detected bounding boxes of each class [44]. Fast R-CNN detects 80x faster than prior R-CNN on a single fish image while increasing mean average precision (mAP) by 11.2% compared to the Deformable Parts Model (DPM) baseline [43].

According to the authors of paper [45], due to the poor image quality and the fish's uncontrolled mobility, traditional feature extraction and object detection techniques as CNN-based algorithms, are not adequate for detecting fish in real underwater conditions. This research suggests a framework for composite fish detection called Composited Fish-Net to overcome this problem and enable efficient fish localization and recognition in complicated underwater environments. The designed composite backbone model (CBres-net) is an improved version of ResNet that aims to capture information on scene change, which is related to the brightness of the image, shape, and orientation of fishes, and seabed structure. The resultant experiments demonstrate that the suggested Composited FishNet's average precision $(AP)_{0.5:0.95}$, average recall $(AR)_{max=10}$, and average precision AP_{50} are, respectively, 75.2%, 81.1%, and 92.8%. The composite backbone network improves the characteristic information utilization and the output of characteristic information for the detected object.

Although the deep learning-based approach is highly effective in order to solve object detection tasks, it has a high computational cost. Xiu Li et al. [14] introduced deep but a lightweight neural network model for detecting fishes in underwater. The ImageCLEF dataset, which contains 24,277 fish photos from 12 classes, attained SOTA accuracy for detecting fish. By utilizing some building blocks, such as concatenated ReLU, HyperNet, and Inception, we alter the structure of convolution layers in comparison to commonly used detection networks, such as Faster R-CNN. The resulting network outperformed the Faster R-CNN network on the same dataset by 7.25%, achieving the top results of 89.95% mAP (mean average precision).

In the automatic localization, classification, counting, and tracking tasks of fish, the research [35] proposed a Mask R-CNN object detection framework. The Mask R-CNN is the expanded version of the Faster R-CNN by adding the new branch for the object mask prediction parallel to the existing branch, which detects the bounding box [46]. The essential component of Mask R-CNN which is missed in Fast/Faster R-CNN, is pixel-to-pixel alignment. The same two-stage technique, with the same first-stage region proposal network (RPN), is used by Mask R-CNN. In the second stage, Mask R-CNN additionally produces a binary mask as output for each RoI in parallel to class and box offset predictions.

To train and verify the model accuracy, a brand-new dataset is presented which consists of labeled images of the fish species indigenous to Southern Africa, the Roman seabream (Chrysoblephus laticeps). The Mask R-CNN model correctly identified and categorized roman seabream on training, validation, and test dataset with mAP_{50} = 80.29%, mAP_{50} = 80.35% and mAP_{50} = 81.45% respectively. The model's success on previously unexplored data shows that it can generalize to new streams of data that are not part of this study [35].

In fact, above mentioned methodologies use the deep learning-based approach in order to solve the underwater fish detection problem. Because of the time-criticality of some tasks, deep learning-based solutions would have limited deployment in some scenarios. The research conducted by Ahmad Salman et al.[47] utilized the probabilistic background modeling for detecting the fishes in complex backgrounds. In the paper, it is mentioned that the Gaussian Mixture Modelling is the efficient method for segmenting fish which is in the foreground, from the background by learning the distribution of pixels in the background. The authors proposed an approach on the base of Gaussian Mixture Models and Pixel-Wise Posteriors for fish detection in complex underwater environments. The outcome of the presented technique is provided using the Complex Background dataset taken from the Fish4Knowledge repository. The F-score generated by their suggested technique is 84.3%, which is noted as the highest result ever recorded on the aforementioned dataset for fish detection. Due to the real-time performance and potential accuracy in dynamic background situations, they used adaptive background subtraction, Grimmson GMM [48] in their method. Gaussian mixture model or GMM is a probabilistic representation of the data distribution using numerous individual Gaussian distributions. Given these features which are the pixel values of a given input, GMM develops a model of the background, where the background is referred to as all objects in the film other than fish. The fish that is needed to be detected is the foreground. The following is the background GMM model [47]:

$$S_M = \left\{ w_j, \mu_j, \sum_j \right\} \tag{2.3}$$

where the jth feature vector's mean and covariance matrices, which correspond to pixel j, are represented by μ_j and $\sum_j$, respectively. The feature vector is the combination of values of each pixel that describes the feature of input data in the following video frames. The authors applied Pixel-wise Posteriors to enhance the segmentation outcomes from the background subtraction in order to address the constraints of adaptive background subtraction. This method is particularly helpful for slow-moving objects, which background subtraction algorithms can only partially identify. By the implementation of the Pixel-wise Posteriors, they achieved 84.28% F-score addition to the background subtraction with GMM, which was 83.26%.

Generally, the use of CNN-based neural networks appears to be a more promising approach for detecting fishes in underwater environments because of their higher accuracy, ability to identify fish in intricate surroundings, and resilience to challenging lighting and water conditions.

3. Methodology

The section methodology is structured as follows. Section 3.1 defines the object tracking problem. The SOTA algorithms utilized for tracking which are the main part of this work explained deeply in Section 3.1.1. Section 3.2 discusses the YOLO detection model as a part of the tracking system. The training of embedding model of tracker for appearance extraction is described in Section 3.3.

3.1 Object tracking problem

Object Tracking is one of the leading problems of computer vision that tries to detect and track objects in the sequences of images. There are challenges in this field which makes it an ongoing research area. There are two types of object tracking methods: single object tracking and multiple object tracking. Single Object Tracking only tracks an individual target object which is given in the first frame of the video and then must track the same object in the next frames. This type of object tracker should be able to track any given object even if there is no classification model given for that object [49].

On the other hand, Multiple Object Tracking (MOT) is more complex. In addition to challenges in single object tracking, MOT needs to track multiple objects in the same category, re-identify the objects when they appear again or terminate when they are out of the vision area of the camera. Furthermore, background clutter, occlusion, and pose changing problems are more complicated compared to tracking single object [50].

3.1.1 Multiple Object Tracking

To conduct the work for solving underwater object tracking, this task is approached as the MOT problem. In this way, it is possible to track many fishes in each frame of the video stream and analyze their behavior individually. As it is described in Section 3.1, tracking is challenging underwater because of the environmental conditions. Being fish as the chosen object also brings difficulties due to their unpredictable movements, interactions with each other, and visual similarities. To handle the mentioned problems the four state-of-the-art tracking algorithms are selected because of their effectiveness and the proven performance in multiple object tracking [51]: DeepSORT, StrongSORT, ByteTrack, and OCSort.

Modern MOT systems generally apply the tracking-by-detection concept: it contains a detection model to locate the target and also, an appearance embedding model for the association of data. In the detection step, targets are localized in the single frame, and in the association step, detected target objects are assigned and linked to the existing trajectories [52].

3.1.2 MOT Problem

The MOT can be seen as the problem of multi-variable estimation. In the given sequence of image, we can define s_t^i to denote i-th object state in the t-th frame, $S_t = (s_t^1, s_t^2, ..., s_t^{M_t})$ to describe the M_t objects states in the t-th frame. The sequential states of the i-th object can be defined as $S_{i_s:i_e} = (s_{i_s}^i, ..., s_{i_e}^i)$, where i_s and i_e are the first and last frame respectively where i target exists and $S_{1:t} = (s^1_, s^2, ..., s_t)$ to define all sequential states of all objects from first to t-th frame.

According to the tracking-by-detection paradigm, we employ o_t^i to describe the observations of the i-th object in the t-th frame, $O_t = (o_t^1, o_t^2, ..., o_t^{M_t})$ to define the all M_t objects observations in the t-th frame and $O_{1:t} = (o_1, o_2, ..., o_t)$ to define all the sequential observations of all objects from first to t-th frame.

The multiple object tracking objective is to obtain "optimal" sequential states of all objects and it can be modeled by applying MAP (Maximum a Posteriori) estimation using the conditional distribution of the sequential states in the given observations:

$$\hat{S}_{1:t} = arg_{S_{1:t}} max P(S_{1:t}/O_{1:t})$$

To solve above mentioned MAP problem, different MOT algorithms can be designed either from deterministic optimization or probabilistic inference perspective [7].

3.1.3 DeepSORT

SORT is a practical approach for the multiple object tracking problem with simple and effective algorithms. In the paper [53], the performance of SORT is improved by adding the appearance information. Because of the extension, it is possible to track the objects during longer periods of occlusions with an effective reduction of the number of identity switches. The more complexity of computation is placed in the offline re-training stage in which deep association metric is learned on the large-scale re-identification dataset of a person. The results of experiments show that the proposed extensions decrease the number of the identity switches by 45%, reaching competitive performance at the high number of frames.

The association between Kalman filter predictions and newly received measurements can be defined by building an assignment problem that can be solved by applying the Hungarian algorithm.

To include the motion information, the Mahalanobis distance between predictions of Kalman states and newly received measurements is used:

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1}(d_j - y_i) \tag{3.1}$$

where (y_i, S_i) is the i-th track distribution projection into measurement space and d_j is the j-th bounding box detection. The Mahalanobis distance includes the estimation uncertainty by measuring the number of deviations between the detection and mean track location. The unlikely associations can be excluded by 95% threshold confidence of the Mahalanobis distance calculated from the x^2 inverse distribution. This decision can be represented as the following indicator

$$b_{i,j}^{(1)} = 1[d^{(1)}(i,j) \leq t^{(1)}] \tag{3.2}$$

that equals to 1 if the association of i-th track and the j-th detection is acceptable. The Mahalanobis threshold for four-dimensional measurement space corresponds to $t^{(1)} =$ 9.4877.

The Mahalanobis distance is an appropriate association metric for the low uncertainty motion. However, in the image-space problem, the state distribution predicted by the Kalman filter is the rough estimation of the object's location. Specifically, the camera motion can cause rapid space displacement which makes the Mahalanobis distance uninformed metric for tacking during occlusions. Because of that authors proposed a second metric for the assignment problem. The appearance descriptor r_j ($\|r_j\| = 1$) is determined for each d_j box detection. Furthermore, the gallery of the last $L_k = 100$ appearance descriptors $R_k = \left\{r_k^{(i)}\right\}_{k=1}^{L_k}$ is saved for each track k. Then the second metric calculates the distance in appearance space between j-th detection and i-th track.

$$d^{(2)}(i,j) = min\left\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in R_i\right\} \tag{3.3}$$

Once more the binary variable is introduced to indicate whether the association is acceptable according to the metric

$$b_{i,j}^{(2)} = 1[d^{(2)}(i,j) \leq t^{(2)}] \tag{3.4}$$

and the separate training dataset is used for finding the threshold for this indicator.

The two metrics are combined with a weighted sum for building association problem

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda)d^{(2)}(i,j) \tag{3.5}$$

If the association is within the gating range of these two metrics, it is called admissible:

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)} \tag{3.6}$$

The hyperparameter λ allows to control the influence of each metric. According to experiments, it is reasonable to set $\lambda = 0$ when there is significant camera motion.

The parameters which are used to run the DeepSORT model are shown below:

$max\_iou\_distance = 0.7$
$max\_age = 30$
$n\_init = 3$
$nms\_max\_overlap = 1.0$
$max\_cosine\_distance = 0.2$
$embedder = "torchreid" (default : "mobilenet")$

3.1.4 StrongSORT

StrongSORT [54] is an improved version of DeepSORT which uses two algorithms: appearance-free link model (AFLink) which is proposed to associate short tracklets into complete trajectories and Gaussian smoothed interpolation for the compensation of missed objects.

In the StrongSORT, two fundamental "missing" problems are discussed: missing association and missing detection. The authors propose an appearance-free link model to accomplish the global association with no appearance information and obtain a good balance between accuracy and speed. Additionally, Gaussian-smoothed interpolation which is based on Gaussian process regression, is proposed for relieving missed detection.

The Optimized Faster R-CNN model is used in DeepSORT as the detector and as the embedding model, it trains simple CNN. Alternatively, StrongSORT applies YOLOX-X for detection, and additionally, a strong feature extractor, BoT, for appearance replaced simple CNN to extract more distinctive features. Even though long-term information can be saved by the feature bank mechanism in DeepSORT, it is sensitive to noise during detection. The feature bank mechanism is replaced with a feature updating strategy for solving this problem. It updates the appearance state of i -th tracklet at frame t which is denoted as e_i^t, an exponential moving average (EMA):

$$e_i^t = \alpha e_i^{t-1} + (1 - \alpha) f_i^t \tag{3.7}$$

f_i^t is the current detection appearance embedding and α is the momentum term which is equal to 0.9.

Multiple benchmarks have camera movements. In StrongSORT, an enhanced correlation coefficient maximization model (ECC) is used for the compensation of camera motion. This technique can compute the global rotation and translation between nearby frames. The performance of warping transformation can be quantified with the following criterion:

$$E_{ECC}(p) = \left\| \frac{\bar{i_r}}{||\bar{i_r}||} - \frac{\bar{i_w(p)}}{\left||\bar{i_w(p)}\right||} \right\|^2 \tag{3.8}$$

where $||.||$ is the Euclidean norm, p is parameter of warping, $\bar{i_r}$ and $\bar{i_w}$(p) are the zero-mean of reference image i_r and warping image i_w(p) respectively. The problem of the image alignment is solved by minimizing E_{ECC}(p) with the presented algorithm of forward additive iterative or inverse compositional iterative. Because of its effectiveness and efficiency, ECC is widely used in MOT tasks for compensating the motion noise caused by camera movement.

In the StrongSORT algorithm, instead of the vanilla Kalman filter, which ignores information on the detection noise scales, the NSA Kalman filter is used. It proposes the

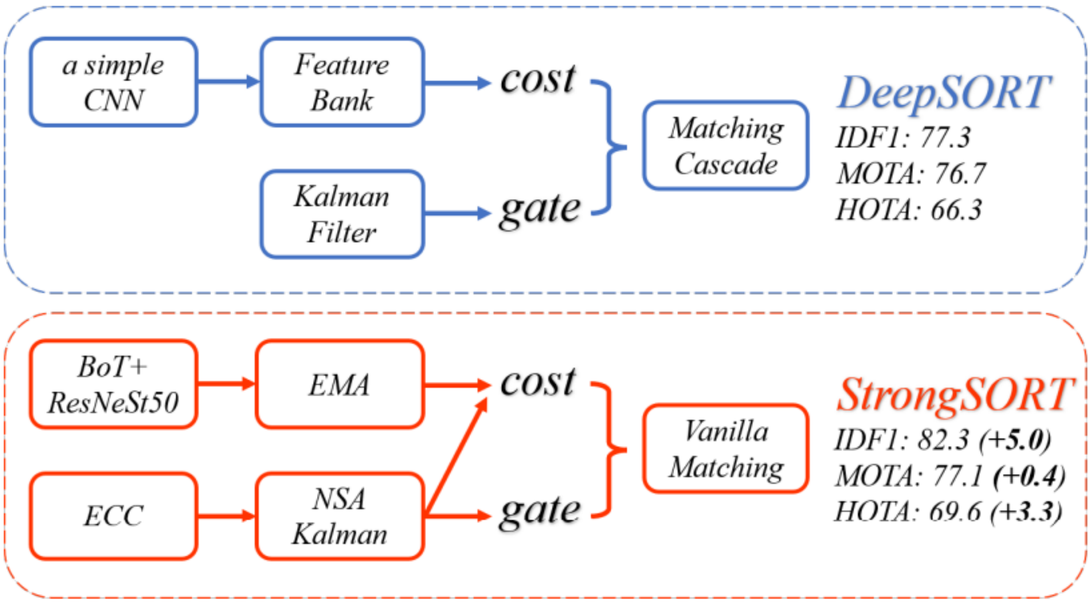


Figure 2. Comparison on the framework and performance between DeepSORT and StrongSORT [54].

below-mentioned formula for adaptive calculation of noise covariance $\tilde{R}_x$:

$$\tilde{R}_k = (1 - c_k)R_k \tag{3.9}$$

R_k is the preset constant covariance for measurement noise, c_k is the confidence score of detection at state k. When it has low noise, the detection has higher c_k which brings the low $\tilde{R}_k$. The lower value of $\tilde{R}_k$ means that the detection will get a higher weight in the step of state update and vice versa. It can help the improvement of the accuracy of updated states.

DeepSORT applies the distance of the appearance feature as the matching cost for the first association stage, which uses motion distance for gate purposes. However, StrongSORT employs both appearance and motion information for solving the assignment problem. The following equation shows the cost matrix C which equals the weighted sum of appearance cost A_a and motion cost A_m [54] :

$$C = \lambda A_a + (1 - \lambda)A_m \tag{3.10}$$

where the weight factor λ is equal to 0.98.

As the tracker algorithm becomes stronger it turns into more robust to the confusing associations. It is simply solved in StrongSORT by replacing the matching cascade in DeepSORT with the vanilla global linear assignment.

Furthermore, the authors proposed using two lightweight, appearance-free, model-independent algorithms which are AFLink and GSI, to solve missing association and missing detection problems. The final algorithm is called StrongSORT++ which combines StrongSORT with these two algorithms.

The StrongSORT algorithm is initialized as shown below in the experiment:

$max\_dist = 0.2$
$max\_iou\_dist = 0.7$
$max\_age = 70$
$max\_unmatched\_preds = 7$
$n\_init = 3$
$nn\_budget = 100$
$mc\_lambda = 0.995$
$ema\_alpha = 0.9$

3.1.5 ByteTrack

ByteTrack [55] compare to other algorithms detects not only objects which have detection with higher scores but also low-scored boxes. The low-scored objects are utilized with the similarities of tracklets to recover real objects.

Most MOT algorithms get identities of objects by associating the boxes which have higher scores from the threshold. The boxes with low scores are thrown away which causes them to miss eligible objects and fragmented trajectories. To solve the mentioned problem, ByteTrack is presented which associates almost every detection box rather than only high-scored ones. It keeps almost all detection boxes and separates them into high and low scores. Firstly, high-score boxes are matched with tracklets on the base of the motion or appearance similarity. Authors adopt the Kalman filter for predicting tracklet locations in the new frame. IoU or Re-ID is used for computing the similarity between the detection box and the prediction box. The second matching is performed using the motion similarity between unmatched tracklets and low-score detection.

The boxes with low confidence sometimes show the existing objects, e.g. occluded objects. The irreversible errors can be caused for MOT because of the filtering out of these objects

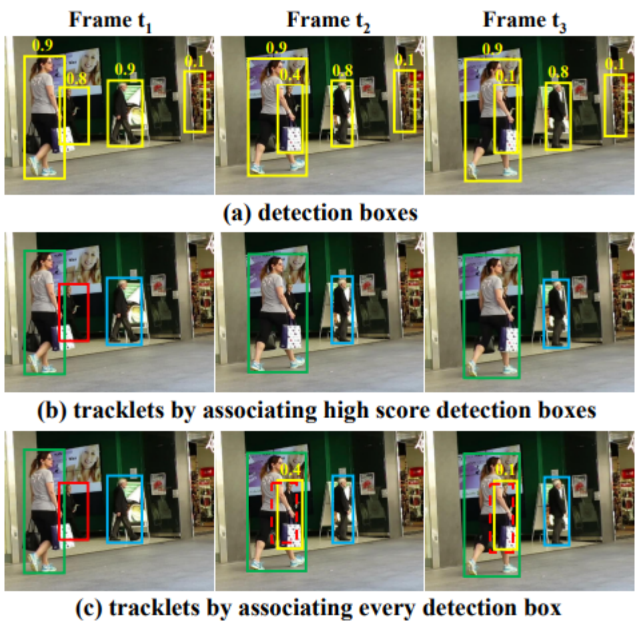

Frame t_1
Frame t_2
Frame t_3
0.9
0.8
0.9
0.1
0.9
0.4
0.8
0.1
0.9
0.1
0.8
0.1
(a) detection boxes
(b) tracklets by associating high score detection boxes
0.4
0.1
(c) tracklets by associating every detection box

Figure 3. Examples of ByteTrack which associates every box of detection [55].

which causes the non-negligible lost detection and fragmented trajectories. Figure 3 (a) and (b) show the same problem. The three tracklets are initialized in the frame t_1 because of having scores higher than 0.5. However, in the frames t_2 and t_3 occlusion happens and the detection score of boxes becomes lower by changing from 0.8 to 0.4 and then 0.1. These low-scored detection boxes are eliminated by threshold and the red tracklet disappears. However, if we take into consideration every detection box, then more false positives will be received immediately, e.g. in the t_3 frame of Figure 6 (a) the right box with 0.1 scores.

In the paper, the authors show that similarity is a powerful point to differentiate the objects and the background in low-scored detection boxes. The two detection boxes which have low scores are matched to the tracklets according to the prediction boxes of the motion model. Simultaneously, the background box is discarded because it does not have matched tracklet.

To fully use the detection boxes in the matching process from high score to low score, the ByteTrack association method is named because each detection box is a tracklet basic unit as a computer byte and every detailed detection is valued by this method.

Initially, 80.3 MOTA, 77.3 IDF1, and 63.1 HOTA are achieved on the MOT17 test set with a running speed of 30 FPS on a single V100 GPU.

The ByteTrack algorithm requires to initialize the below-mentioned parameters in order to run tracking and they have the following values in the conducted experiment:

$track\_thresh = 0.45$
$match\_thresh = 0.8$
$track\_buffer = 25$
$frame\_rate = 30$

3.1.6 OCSort

The goal of the research [56] is to develop a motion model based multi-object tracking method that can deal with occlusion and nonlinear motion. Many of the motion model based algorithms suppose that tracking objects have a constant velocity in a time period which is the linear motion assumption. In actuality, this presumption might not be true which could give inaccuracies in the result. The authors offer a motion model which addresses some of the limitations of current approaches to enhance tracking performance, particularly in the occlusion conditions.

The most popular filtering-based solution for multiple object tracking is SORT which uses Kalman filter in order to predict object states and the function of linear motion to transition between time steps. However, the filter posteriori parameters can not be updated by SORT and it is not robust to the nonlinear motion. In this study, three limitations of SORT -are noted: noise sensitivity, estimation-centric methodology, and error accumulation over time.

This research suggests two primary innovations—Observation-Centric Re-Update (ORU) and Observation-Centric Momentum (OCM)—to overcome these restrictions. During times of lost tracking, ORU employs object state observations to correct accumulated errors. The cost matrix for association in OCM takes track direction consistency into account. The suggested technique, called Observation-Centric SORT (OC-SORT), increases robustness in occlusion and nonlinear motion conditions while offering real-time tracking capabilities [56].

Observation-centric Re-Update (ORU)

In real-world circumstances, even if items tracked by the SORT approach are re-associated after going untracked for a while, they may still go lost again. This is because of the temporal error magnification that has caused the Kalman filter (KF) parameters to already deviate too much from the ideal values. In the research, Observation-Centric Re-Update (ORU) technique is proposed to solve this problem. By backchecking the time the object was lost and re-updating the KF parameters based on "observations" from a virtual trajectory, ORU lowers the accumulative error. By using the observations of the steps that start and end the untracked interval, the virtual trajectory is created. For instance, the virtual trajectory is represented by the denoting the observation last seen before started to be untracked and the observation which triggers re-association by z_{t_1} and z_{t_2} respectively:

$$\tilde{z}_t = Traj_{virtual}(z_{t_1}, z_{t_2}, t), t_1 < t < t_2 \tag{3.11}$$

Then, the predict and re-update loop run along the $\tilde{z}_t(t_1 < t < t_2)$ trajectory. The operation of re-update is

$$re - update \left\{ \begin{array}{c} K_t = P_{t\backslash t-1} H_t^T (H_t P_{t\backslash t-1} H_t^T + R_t)^{-1} \\ \hat{x}_{t\backslash t} = \hat{x}_{t\backslash t-1} + K_t(\tilde{z}_t - K_t x_{t\backslash t-1}) \\ P_{t\backslash t} = (I - K_t H_t) P_{t\backslash t-1} \end{array} \right\} \tag{3.12}$$

The update will no longer be suffered from the accumulated error by update because pattern of motion anchored by the last-seen and the most recent real observations is matched by observations on virtual trajectory. The proposed technique is known as Observation-centric Re-Update. It functions as a separate stage outside of the predict-update loop and is only active when a track is brought back online after a time of no observations.

The parameter which is required to run OCSort algorithm has given values as shown below:

$max\_age = 30$
$min\_hits = 3$
$iou\_threshold = 0.3$
$delta\_t = 3$
$asso\_func = "iou"$
$inertia = 0.2$

Observation-Centric Momentum (OCM)

In the research, it is discussed that the motion can be approximated as linear in a short period. The consistent motion direction is required by linear motion assumption but the utilization of consistency of direction is prevented by noise. To calculate the motion direction, it is needed to have states of the object on two steps where the time difference is Δt. The estimator is sensitive to state noise, therefore if Δt is little, the velocity noise would be substantial. Due to the temporal error magnification and the falsification of the assumption of linear motion, if Δt is large, the noise of direction estimation may also be significant. The authors propose using state observations rather than state estimations to reduce the noise of motion direction computation and introduce the term of its consistency to aid the association because state observations do not suffer from the problem of temporal error magnification that state estimations do.

Given N existing tracks and M detections on the upcoming time step, the association cost matrix for the new term is formulated as follows:

$$C(\hat{X}, Z) = C_{IoU}(\hat{X}, Z) + \lambda C_v(Z', Z) \tag{3.13}$$

where the set of estimation of object is denoted as $\hat{X} \in \mathbb{R}^{N\times 7}$ and the set of new time step observations is $\hat{Z} \in \mathbb{R}^{M\times 5}$. λ is weighting factor. Z' contains trajectories of all existing tracks' observations. $C_{IoU}(.,.)$ determines the negative pairwise IoU and $C_v(.,.)$ determines the consistency of directions between Θ^{track} which is linking the two observations on existing track and $\Theta^{intention}$ which is linking historical and new observation of track. Θ consists of all the pairs of $\Delta\Theta = \left|\Theta^{track} - \Theta^{intention}\right|$.

In addition to ORU and OCM, authors also believe that experimentally verifying a track's most recent presence can prevent it from being lost. As a result, the heuristic Observation-Centric Recovery (OCR) method is employed. Following the customary association stage, OCR will begin a second attempt to associate the last observation of unmatched tracks to the unmatched observations. It can deal with the situation where an object stops or is blocked for a brief period [56].

The OCSort algorithm has experimented on multiple datasets such as MOT17 and MOT20. We can say from the benchmark results that OCsort outperforms compare to the other SOTA methodologies.

3.2 Enhancing Object Tracking through YOLO-based Detection

The tracking-by-detection approach is chosen for solving the tracking problem in an underwater environment, and YOLO is considered an appropriate detection model for this purpose because of being a real-time and more accurate model which makes it suitable for tracking fast-moving objects. In this thesis, transfer learned YOLOv7 model on fish dataset is used which is received from the Environmental Sensing and Intelligence Group of Tallinn University of Technology. The model is trained on the 74043 fish images and received F1 score of 0.95, Precision 0.967 and Recall 0.933 [57]. YOLOv7 outperforms other detection algorithms with its speed and accuracy and achieves SOTA performance [58].

In the research, the authors proposed a new real-time object detection architecture and an associated model scaling technique. They discovered the replacement problem for the module of re-parameterize and the allocation difficulty for the dynamic label assignment during the study process. The suggested method named is the trainable bag-of-freebies to address the issue and improve object detection precision. Based on that, the YOLOv7 series of object detection system is created, which produces cutting-edge results.

3.3 Training embedding model for appearance extraction

As it is discussed in Section 3.1.1, the algorithms like StrongSORT take into account the appearance of tracklets. It utilizes embedding models for extracting the appearance features of objects. In the original paper, the authors of the StrongSORT algorithm propose to use the BoT model as a feature extractor instead of the simple CNN model trained for the DeepSORT model. In order to run the StrongSORT algorithm, it is possible to load the weights of pretrained models as MobileNET, ResNET50, and OSNet which are trained on the general dataset. In this thesis work, the OSNet model is used as an embedding model. OSNet is a re-ID CNN model which stands for omni-scale network [59]. It can be needed to train the feature extractor model on the basis of the fish data because of feature difference from general domain objects. The training process of the model is following:

1. preparation of the dataset by cropping the fishes from images and saving the path of images together with the object ID and camera ID in a CSV file. The example format from a dataset is shown in Table 1.

	image_pth	pid	camid
0	datasets/images/000.jpg	0	0
1	datasets/images/001.jpg	0	0
2	datasets/images/002.jpg	0	0
3	datasets/images/003.jpg	0	0

   Table 1. The example dataset for the training of embedding model.

   The ID of fishes is saved in such a way that same fishes extracted each frame of video receives same ID and each different fish in the videos get different ID. The camera ID for all videos are chosen as zero.
   The cropped fish images from underwater videos are illustrated in Figure 4.
2. The dataset is divided into the train, test, and query subdatasets and managed by the ImageDataManager() function of the torchreid library. Torchreid is a deep learning-

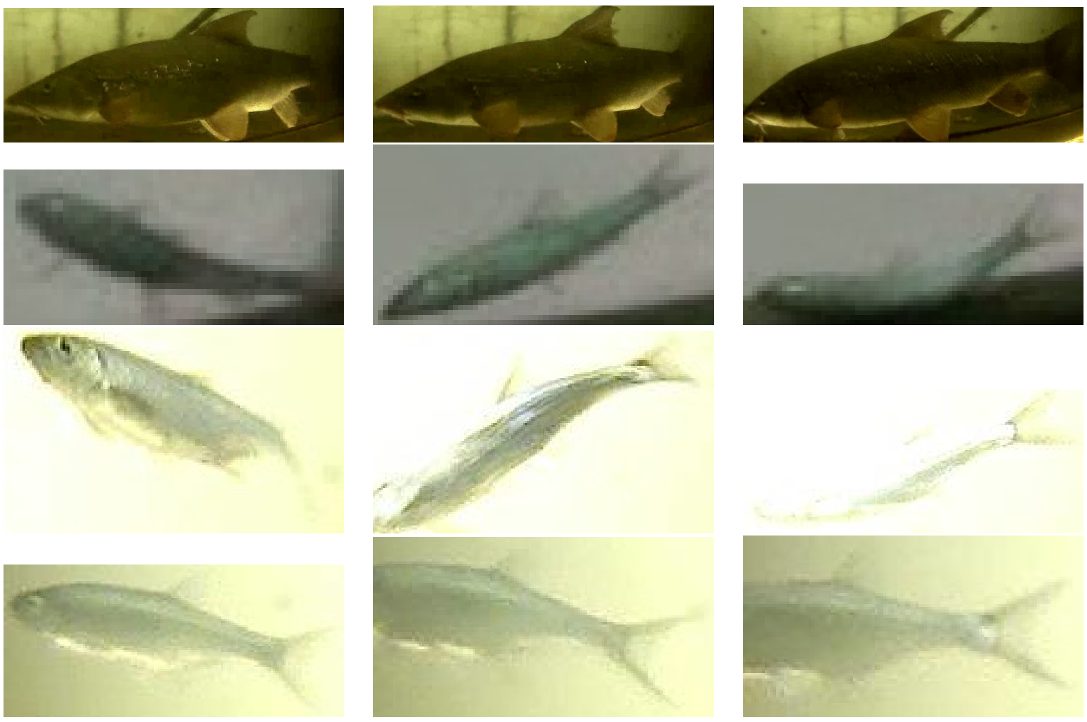


Figure 4. Cropped fishes from BfG underwater fish videos [17] using Python script for the training of embedding model.

based re-identification library that is developed in PyTorch for the ICCV'19 project [60]. It contains many features as access to the pretrained reID models, advanced techniques for training, and visualization tools. The managed dataset specifications are demonstrated below:

```
----------------------------------------
subset   | # ids | # images | # cameras
----------------------------------------
train    |     7 |      666 |         1
query    |     2 |       40 |         1
gallery  |     2 |      187 |         1
----------------------------------------
```

3. The 'osnet_ain_x1_0' model is initialized and trained by using builtin functions of Torchreid.

In fact, embedding models for extracting the feature appearances of tracklets are trained on the reID datasets as Market1501 [61]. However, it is possible to train the model on the base of the custom dataset for specific tasks by using the above-mentioned steps.

3.4 Evaluation metrics

Choosing the evaluation metrics has higher importance for not loosing any important information about each algorithm. The selected evaluation metrics are following: IDF1, IDsw, Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP).

To increase the overall number of frames in which corresponding tracks overlap, or the Identification True Positives (IDTP), IDF1 builds correlation between full tracks [62].

$$IDF1 = M_{-1}(IDR, IDP) = IDTP / \left[\frac{1}{2}(N + \hat{N})\right] \tag{3.14}$$

N - number of boxes in tracks of ground truth
$\hat{N}$ - number of boxes in predicted tracks
M_p - the p-mean
IDR - recall
IDP - precision

IDsw is the Identity Switch which happens when a tracker incorrectly switches the identities of objects or when a track is lost and then reinitialized with a new identity [63]. An IDsw is formally defined as a true positive with a predicted ID that is distinct from the predicted ID of the prior true positive (which has the same ground truth ID). IDsw determines only associated errors comparison to the single prior true positive and do not consider where the same predicted ID swithces to another ground truth ID which is called ID Transfer.

The MOTA demonstrates that if the tracker algorithm found the correct object in a frame. It is represented as below mentioned function:

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{3.15}$$

where m_t, fp_t and mme_t are number of misses, false positives and mismatches respectively.

The Multiple Object Tracking Precision (MOTP):

$$MOTP = \frac{\sum_{i,t}(d_t^i)}{\sum_t c_t} \tag{3.16}$$

where d_t^i is the total error between matched pairs of object-hypothesis in all frames, c_t is the total number of matches [64].

4. Results

This section presents the results of the tracking methods employed for tracking fish in an underwater environment with the created MOT dataset.

The experiment is conducted on two datasets, one containing 87 frames and the other 246 frames. The detailed information about dataset is also given in Section 5.1. The first dataset contains frames from a single video that has three fishes on screen and 252 annotated bounding boxes. The second dataset consists of frames from three videos which have three fishes in the first and second video, and two fishes in the last video, and overall 430 annotated bounding boxes. The ground truth positions of each fish species are annotated manually using CVAT in MOT16 format, which is mentioned as follows [65, 66]:

<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>

Example rows for such annotation format are shown below from the dataset used in evaluation:

1, 1, 603.0, 401.9, 105.9, 46.0, 0.9, -1, -1, -1
2, 1, 597.0, 411.9, 99.9, 45.0, 0.8, -1, -1, -1
3, 1, 583.0, 420.9, 114.9, 44.0, 0.8, -1, -1, -1
4, 2, 715.0, 43.9, 47.9, 21.0, 0.7, -1, -1, -1

Table 2. illustrates the description of each element in the data format which is valid for the result of the tracker and also, ground truth files. The ground truth annotation data and also, the tracking results are saved in separate .txt files according to the above-mentioned format.

The experiments are done on the state-of-the-art MOT trackers explained in Section 3.1. The tracking algorithms require inputs to be executed, which is true for all four models investigated in this thesis work. The needed data must include the bounding box parameters(commonly, x and y coordinates of starting point together with width and height), confidence and class of detection, and also, the frame. It can be given in different formats depending on the implementation of the algorithm:

```
    <x>, <y>, <w>, <h>, <conf>, <cl>
```

Position	Name	Description
1	Frame number	shows that at which frame of stream object exist
2	Identity number	each tracked object has unique identity number
3	Bounding box left	top-left corner coordinate of the bounding box
4	Bounding box top	top-left corner coordinate of the bounding box
5	Bounding box width	width in pixels of bounding box
6	Bounding box height	height in pixels of bounding box
7	Confidence score	show how confident is detector that this instance is needed object. For the result and also, ground truth, it describes whether detection is considered.
8	x	3D x position of detected object in real-world coordinates (-1 in case it is not available)
9	y	3D y position of detected object in real-world coordinates (-1 in case it is not available)
10	z	3D z position of detected object in real-world coordinates (-1 in case it is not available)

Table 2. Data format of detection and annotation file [65].

The below-mentioned code is a part of the tracker.py file which executes the tracking algorithm in the software. The track() (Fig. 5) function takes the input frame to run the YOLO detection model, which returns ’x’, ’y’, ’w’, ’h’, ’conf’, and ’cl’ data of each detected object in order to give as an input to the initialized DeepSORT model using update_tracks() function together with the frame. At the end of the track() function, it returns the boxes and corresponding IDs of each tracked object.

Each method is evaluated using IDF1, IDsw, MOTA, and MOTP metrics. For the calculation of metrics, the py-motmetrics Python library is used. In comparison to the MOT Challenge benchmark computation of metrics, the library does not calculate MOTP as a percentage and uses the original formula [64]. It can be converted by computing (1-MOTP)*100.

The initial experiment is conducted on the first dataset, which consists of 87 frames. The comparative results of trackers are shown in table 3. According to the table, DeepSORT, StrongSORT, ByteTRACK, and OCSort, all have an identical IDF1 score of 0.5898 and IDsw value of 8. This shows that for identifying and tracking fish in a certain scene, all four techniques are equally effective. The MOTA and MOTP scores, however, are different from one another. Among the four approaches, OCSort has the highest MOTA score (0.5940), indicating that it has the fewest false positives, false negatives, and identity switches. The MOTA score for the other three approaches is 0.5912. Lower MOTP scores imply better tracking accuracy since they reflect the average distance between predicted and actual object positions. The table shows that all four approaches have a MOTP score

```
def track(frame):
    detection = model(frame)
    tracker = DeepSort(max_age=30)
    num= detection.xyxy[0].numpy().shape[0]
    boxes=[]
    id=[]
    detections=[]
    for i in range (num):
        x= int(detection.xyxy[0].numpy()[i][0])
        y = int(detection.xyxy[0].numpy()[i][1])
        w= int(detection.xyxy[0].numpy()[i][2])
        h=int(detection.xyxy[0].numpy()[i][3])
        conf = detection.xyxy[0].numpy()[i][4]
        cl = int(detection.xyxy[0].numpy()[i][5])
        detection= ([x, y, w, h], conf, cl)
        detections.append(detection)
    tracks = tracker.update_tracks(detections, frame= frame)

    for track in tracks:
        boxes.append(track.to_tlwh())
        id.append(track.track_id)
    return boxes, id
```

Figure 5. The track() function which contains the full process of tracking.

of approximately 0.17. Overall, OCSort seems to have the best tracking accuracy, with the other three approaches closely following in the first experiment.

Method	IDF1	IDsw	MOTA	MOTP
DeepSORT	0.5898	8	0.5912	0.1707
StrongSORT	0.5898	8	0.5912	0.1707
ByteTRACK	0.5898	8	0.5912	0.1707
OCSort	0.5940	10	0.6111	0.1698

Table 3. Evaluation results of MOT methods in the first dataset.

The next experiment is done by utilizing a second dataset which contains 246 frames. Table 4 indicates that the IDF1 scores for the four approaches, DeepSORT, StrongSORT, ByteTRACK, and OCSort, range from 0.6795 to 0.6844. The IDsw, MOTA, and MOTP scores, however, are not all the same. Among the four algorithms, OCSort has the fewest identity switches (118), according to its lowest IDsw value. By contrast, ByteTRACK has an IDsw value of 119, whereas DeepSORT, StrongSORT, and SORT have IDsw values of 121. We can see that for MOTA, DeepSORT, StrongSORT, and OCSort all of their scores are 0.4744. With a slightly lower MOTA score of 0.4720 than the other three approaches, ByteTRACK has a little greater rate of false positives, false negatives, and identity switches. Finally, all four approaches have comparable MOTP scores, which range from 0.1562 to

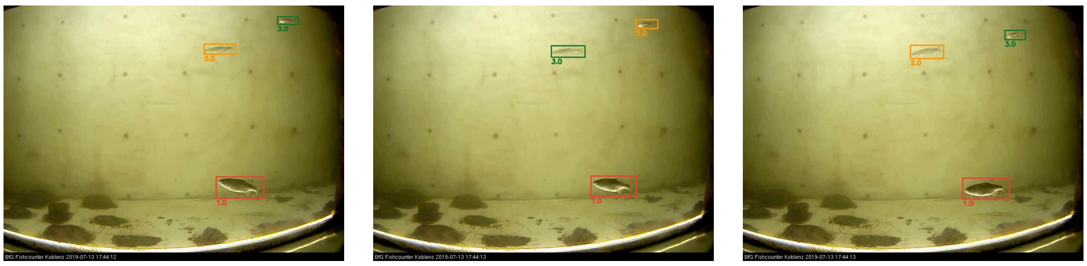


Figure 6. Identification switch in OCSort.

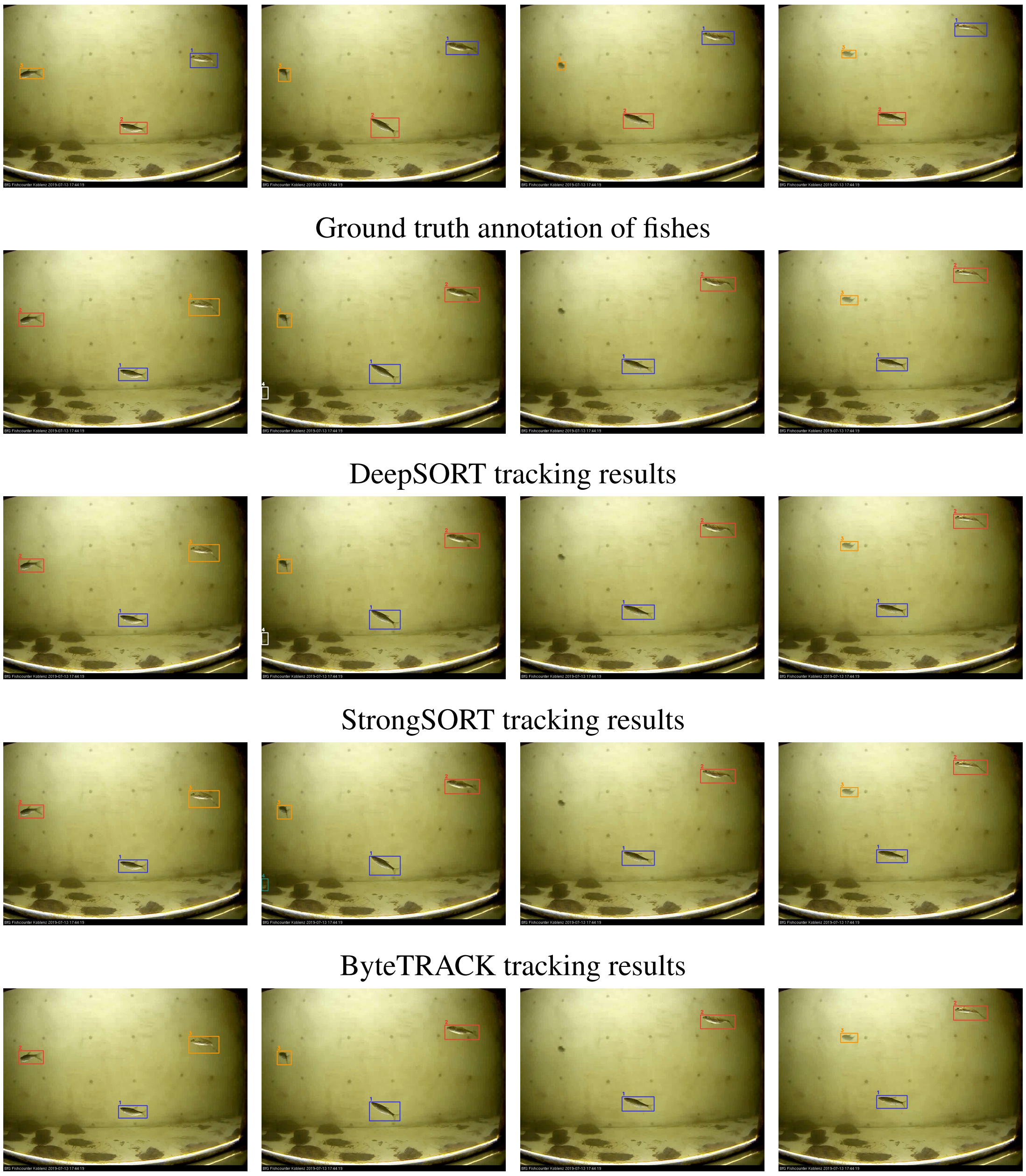
0.1580. In comparison to the other three approaches, although OCSort has low IDF1 score, it can be concluded that OCSort can still be considered as a viable approach considering overall performance.

Method	IDF1	IDsw	MOTA	MOTP
DeepSORT	0.6844	121	0.4744	0.1580
StrongSORT	0.6844	121	0.4744	0.1580
ByteTRACK	0.6819	119	0.4720	0.1569
OCSort	0.6795	118	0.4744	0.1562

Table 4. Evaluation results of MOT methods in the second dataset.

Although the fish tracking in some frames have issues in some attempted methods which are identified by comparing with the ground truth data (Fig. 7), it must be noted that these problems do not present in all frames. Furthermore, some racking methods are able to track the fish successfully in that frame without encountering issue. As it is seen from pictures, DeepSORT and StrongSORT detects the fourth fish in the second frame which is not available in ground truth dataset and neglected by ByteTRACK and OCSort. However, in the third frame, all four algorithms missed one of the fish, which is identified by ID 3, because of its orientation, and it is recovered in the fourth frame.

In the first dataset, the results reveal that all four techniques are equally effective at recognizing and tracking fishes, with OCSort having the greatest MOTA score and the fewest identification switches. In the second dataset, all four techniques got comparable MOTP scores, with OCSort outperforming the others with respect to MOTA, MOTP score and the smallest number of identity switches. In both studies, OCSort appears to have higher tracking performance.


Ground truth annotation of fishes
DeepSORT tracking results
StrongSORT tracking results
ByteTRACK tracking results

OCSort tracking results

Figure 7. Comparison of tracking results with ground truth.

5. Implementation

After the development and evaluation of model, Artificial Intelligence Lifecycle includes deployment of model. In this paragraph, implementation of the tracking algorithm is broadly discussed. The structure of this section is as follows. Section 5.1 lists all the data used in the different of this thesis work. The architecture and the structure of developed web application and API are explained in Section 5.2 and 5.3 respectively meanwhile Section 5.4 describes the software dependencies used in development.

5.1 Data

The data which is used for the thesis is received from The German Federal Institute of Hydrology (BfG) [17]. As it is described in Section 4, the two datasets are utilized for evaluating the performance of tracking algorithms. The first dataset which has 87 frames received from single video and it is also part of second dataset. The second dataset consist of 246 frames from three different videos. In order to annotate all frames, Computer Vision Annotation Tool (CVAT) [67] is used and exported in MOT format.

Testing the performance of created application also conducted using the videos which contain complex frames to analyze. They are video frames which have many fishes but not used during the evaluation. One of the main reason is annotation of video frames is challenging task to do manually when there are many small fishes in blurry video. It can be seen clearly in Figure 8. Furthermore, beside of above mentioned evaluation dataset, there is a prepared dataset using Python script for training of embedding model which consists of the cropped images of fishes from separate single fish videos and it is described in Section 4.

5.2 Web Application Architecture

The creation of software that can quickly receive, evaluate, and give tracking information from underwater video footage is one of the main objectives of this thesis. In order to achieve this goal, a web application and API are developed with the software architecture shown in the Figure 9. A Flask web application that uses a PyTorch model to analyze received video from user and outputs both the analyzed video and JSON data. It also has a REST API that can be used to communicate with the app directly without going via the frontend.

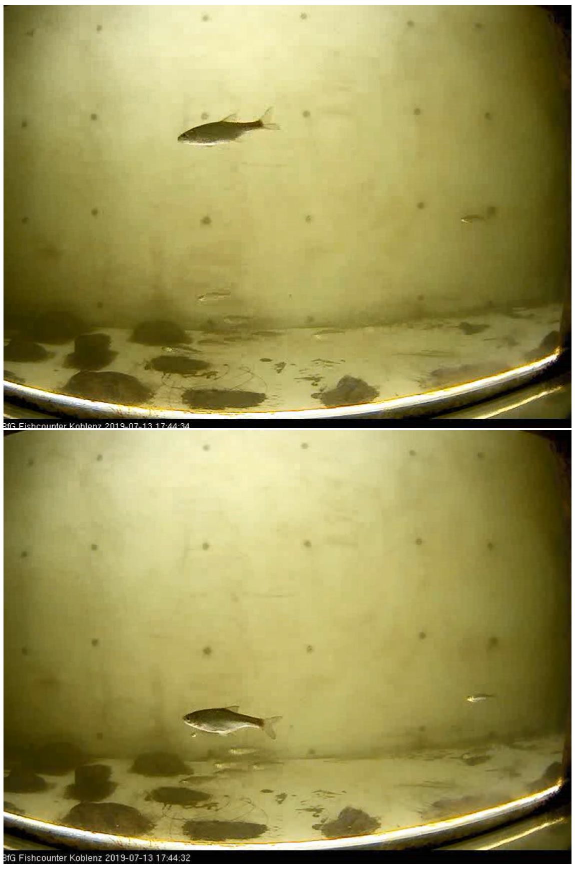

BfG Fishcounter Koblenz 2019-07-13 17:44:34
BfG Fishcounter Koblenz 2019-07-13 17:44:32

Figure 8. Multiple fish appearance in challenging view.

The architecture of the web application:

1. User uploads video to the Flask web application.
2. Flask server receives the video, saves in runtime and sends it to the Pytorch model and tracker algorithm for analysis.
3. The detector and tracking algorithm processes the video and generates tracking information.
4. Information of tracklets are saved in JSON format and new video with tracking information overlay is generated on the base of it.
5. After end of the video processing, the output video and JSON file are downloaded.

For implementing this architecture, the following components are used:

- Frontend: HTML/CSS/Jinja2, which allows users to upload the underwater video file and demonstrates the analyzed video with overlaid tracking information.
- Flask server: Flask is a lightweight micro framework for web application which is written in Python. Flask allows to create one-page applications and also, scale them and develop larger applications without issues [68]. As a web framework, Flask will

be used because of the API support as well. API is a set of functions to provide interactions between computer programs and exchange of information. REST is an architectural style that is applied for design of APIs. Web services called "RESTful" which have REST API [69].

- Pytorch and OpenCV: The uploaded video is handled by OpenCV in order to get the frames, overlay tracking results, and save new video stream. The YOLO model is in Pytorch format and Pytorch framework allows us to load model and run prediction for detecting the fishes on the received video frames which is used by tracker algorithm.
- Database: In fact, current application does not have database and saves the uploaded view in runtime. It is possible to integrate database for saving the tracking results of users.

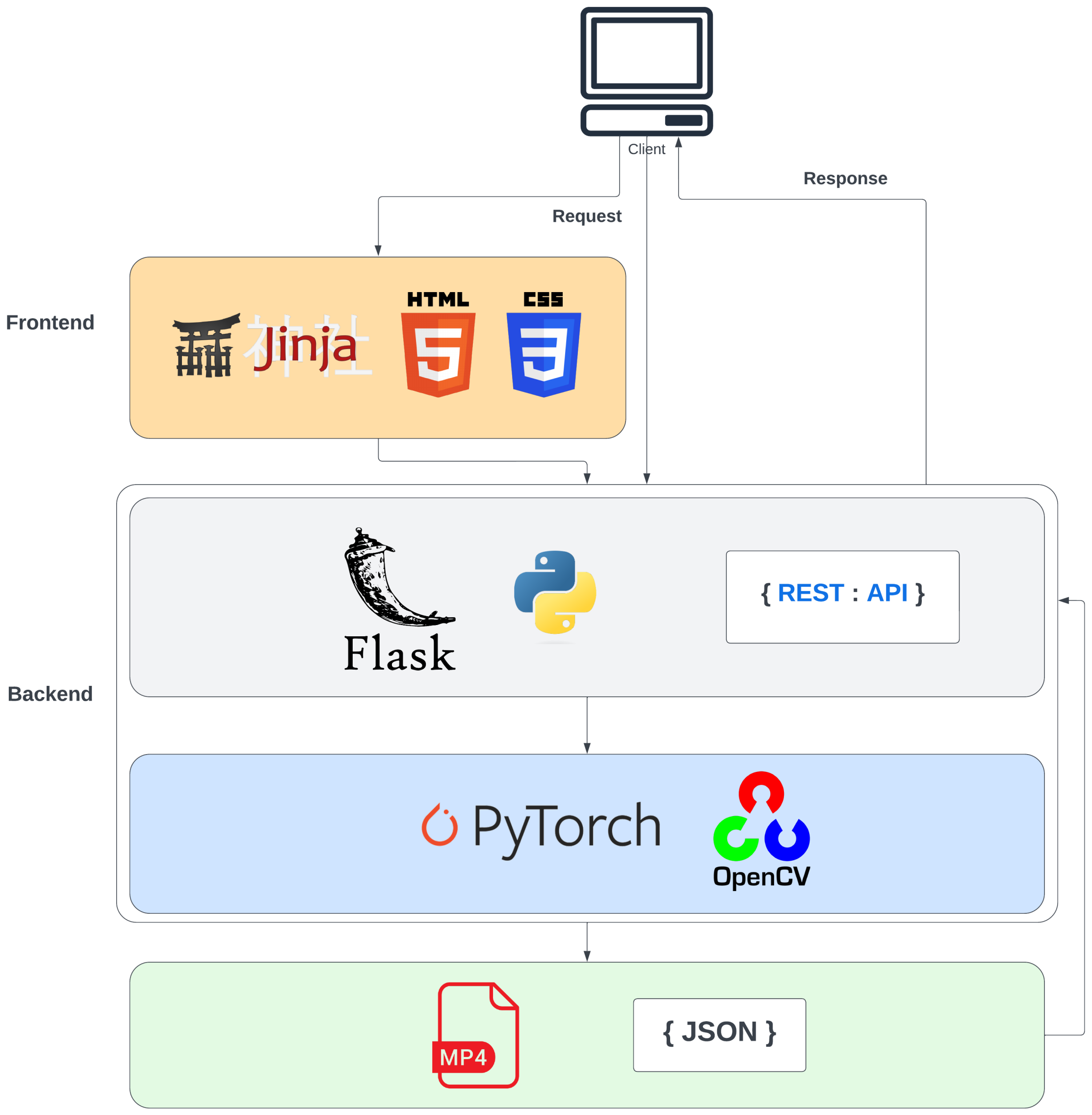


Figure 9. Architecture of fish tracker software.

5.3 RESTful web application

As it is described in the Section 5.2 the developed web application is capable of the video processing for tracking fish species. The file structure of a Flask application created in the Python environment, along with an overview, is presented below.:

```
tracker_app/
|
+-- venv
+-- templates/
|    |
|    +-- index.html
|    +-- video.html
+-- app.py
+-- tracker.py
```

- ’app.py’: This is the main file of the Flask application that contains the route of the application. It has three routes, ’/run’, ’/’, and ’/video_feed’.
- ’templates’ folder: This folder contains the HTML templates that the Flask application uses to render web pages.
    1. The ’index.html’ template contains a simple form for uploading an image file and submitting it to the server.
    2. The ’video.html’ template displays the video overlaid with tracking boxes and IDs on the page.
- ’tracker.py’: This file contains the function ’track()’ which is used for detecting fishes and handling the tracking algorithm. The function receives frames, bounding box, and class names and returns the IDs and bounding box coordinates of tracked fishes.

The overview of how the Flask application works:

1. When a user visits the ’/’ route, they are presented with the ’index.html’ page where they can upload a video file (Figure 13).
2. When the user uploads a video file, the application saves the video file to a temporary file and then renders the ’video.html’ and passes path of the file to the video.html template.
3. ’video_feed/’ (Fig. 11) route handles the video stream page which takes path to the file from ’url_for’ function used in the ’img’ tag of ’video.html’ (Fig.10) template.

```
<body>
<h1>Fish Tracking Video Stream</h1>
<div class="container">

    <img src="{{ url_for('video_feed',
    video_path=video_path )}}" width="50%">

</div>
</body>
```

Figure 10. HTML code for video stream page of web application

```
@app.route('/video_feed', methods=['GET'])
def video_feed():
    # Get the temporary file path
    video_path = request.args.get('video_path')
    # Open the video file
    cap = cv2.VideoCapture(video_path)
    fps = cap.get(cv2.CAP_PROP_FPS)
    # Check if video was opened successfully
    if not cap.isOpened():
        return redirect(url_for('index'))
    # Set the response header
    return Response(generate_frames(cap=cap, fps=fps),
    mimetype='multipart/x-mixed-replace; boundary=frame')
```

Figure 11. 'video_feed/' route for streaming tracking video

4. It uses the generate_frames() function which utilizes track() function. At the result, it generates tacking data from video frames, saves in the JSON format and yields each tracking frames as a binary data stream.
5. The response header is then set to 'multipart/x-mixed-replace; boundary=frame', and the frames of the tracking video stream are returned as a 'Response' object (Figure 13).
6. After completion of the tracking process in all frames of uploaded video, the JSON and tracking video files are downloaded.

In order to get access to the tracking model through Rest API, user needs to send 'GET' request to the '/run' endpoint with the URL of the video. The description of API is shown below.

Endpoint URL: http://localhost:5000/run

HTTP Method: GET
Request Parameters: video_link Example request: GET http://localhost:5000/run?video_-link=<video_link_value>

The format of JSON output for Web application and API is same:

```
{
    "<frame_num_1>": [
        [<start_point>, <end_point>, "<id_1>"],
        [<start_point>, <end_point>, "<id_2>"],
    ],
    "<frame_num_n>": [
        [<start_point>, <end_point>, "<id_1>"],
        ...
        [<start_point>, <end_point>, "<id_m>"]
    ],
    ...
}
```

The key '<frame_num_n>' represents the number of each video frame where fish is detected. The elements of value '[<start_point>, <end_point>, "<id_m>"]' are the starting point, ending point of bounding box and the ID of tracked fish respectively.

5.4 Software Dependencies

The software is implemented as a Flask-based web application running in a Python 3.10 environment. Some software dependencies are required for the application to function properly. The Flask framework (version 2.2.2) provides packages to build the backend of web application. OpenCV (version 4.6.0) is an open-source library which consists of hundreds of computer vision algorithms and it is used for handling the video file and its frames in this thesis work. PyTorch is the machine learning framework that is also renowned in the research community because of being more flexible in building new approaches. The torch library (version 1.13.0+cpu) is utilized for loading the YOLO model and running detection on the frames of underwater video. For application of tracking algorithms in the task, two approaches are used. Firstly, deep_sort_realtime library (version 1.3.2) is used for DeepSORT method which contains all necessary calculations such as Kalman Filtering in order to do real-time tracking of multiple objects. Secondly, the YOLO provides packages in order to use ByteTrack, OCSort, and StrongSORT algorithms but it is needed to do changes over them before use. Furthermore, json library is applied for writing JSON data. Meanwhile, during the conducting experiments, py-motmetrics library (version 1.4.0) is installed which has a Python implementation of MOT metrics, is utilized for evaluation of tracking results.

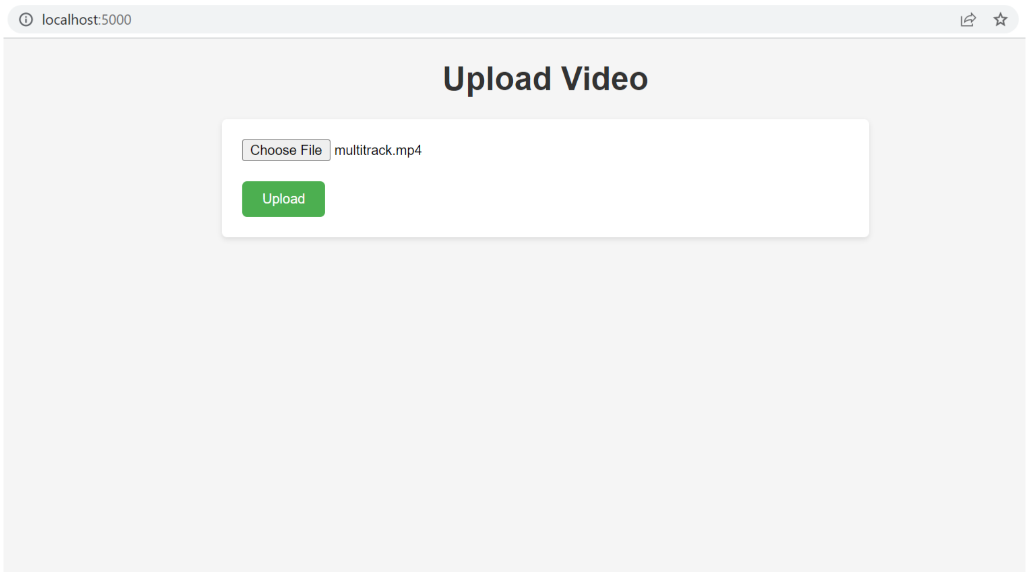


Figure 12. User interface for uploading video to track.

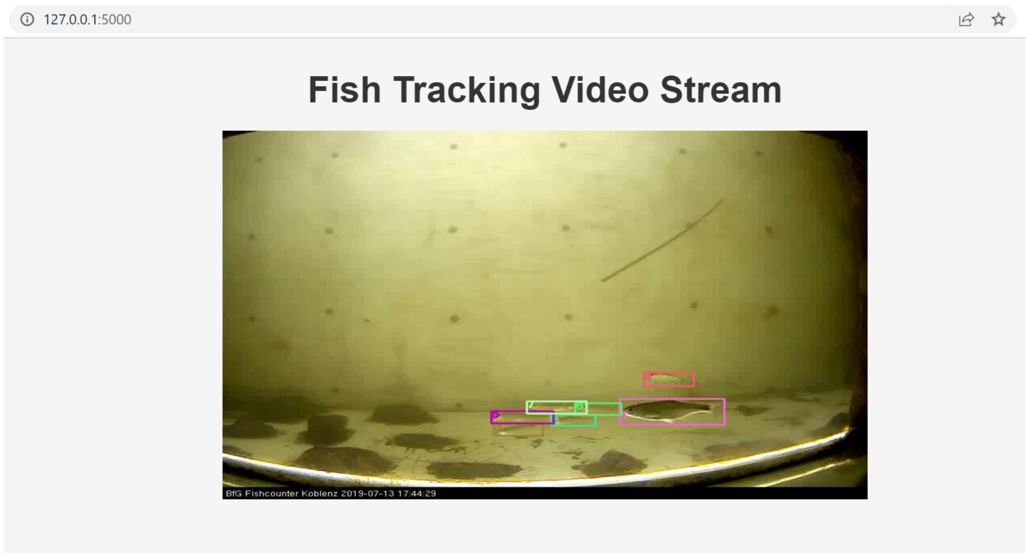

127.0.0.1:5000
Fish Tracking Video Stream
BfG Fishcounter Koblenz 2019-07-13 17:44:29

Figure 13. The screen for object tracking stream.

6. Summary

Underwater object tracking is a difficult problem that has attracted the interest of researches due to its vast range of applications in marine biology, oceanography, and underwater robotics. A detailed literature review was conducted in this master thesis to investigate the state of the art methodologies in underwater object tracking. The study examined the specific models and approaches tried to address the problems related to the underwater environment as well as the characteristics of the objects being tracked. Furthermore, this thesis focuses on the evaluation and deployment of multiple object tracking (MOT) algorithms to solve the underwater object tracking problem.

This thesis presents a comparative evaluation of different multi-object tracking (MOT) algorithms for underwater scenarios. The research involved preparation of MOT dataset on the basis of underwater fish video data received from [17] and evaluation of four state-of-the-art MOT algorithms: DeepSORT, StrongSORT, OCSort, and ByteTRACK. The performance of each algorithm was measured by precision, accuracy, F1 score and the number of identification switches. The results showed that OCSort achieved the best performance among the four algorithms, with low identification switches and high precision and accuracy.

Furthermore, this thesis provides an in-depth examination of the process of applying the evaluated algorithms to the problem of underwater object tracking. The investigation took into account several processes, such as input data preparation, model training and evaluation, and identified the primary obstacles associated with implementing tracking algorithms in the underwater environment. The study gives useful information for tracking model selection and application on the specific domain. Finally, this thesis presents a web application and API that demonstrate the practicality of the chosen tracking algorithms and makes the solution more accessible. The application has the potential to be expanded by including tracking into live stream fish videos and developing a database from which users may obtain the results.

In conclusion, this research offers a thorough overview of the difficulties in underwater object tracking as well as a detailed analysis of the particular models and approaches developed to deal with these difficulties. Future research in this area will be well-supported by further investigation with extended dataset and optimization of tried algorithms, and the operational considerations involved in using them. The results of this study have a potential impact on underwater object tracking research and improve the state-of-art in this area.

References

[1] Gabrielle Canonico et al. "Global Observational Needs and Resources for Marine Biodiversity". In: *Frontiers in Marine Science* 6 (2019). ISSN: 2296-7745. DOI: 10.3389/fmars.2019.00367. URL: https://www.frontiersin.org/articles/10.3389/fmars.2019.00367.

[2] Jacopo Aguzzi et al. "Coastal observatories for monitoring of fish behaviour and their responses to environmental changes". In: *Reviews in fish biology and fisheries* 25 (2015), pp. 463–483.

[3] Vishnu Kandimalla et al. "Automated Detection, Classification and Counting of Fish in Fish Passages With Deep Learning". In: *Frontiers in Marine Science* 8 (2022). ISSN: 2296-7745. DOI: 10.3389/fmars.2021.823173. URL: https://www.frontiersin.org/articles/10.3389/fmars.2021.823173.

[4] Weiran Li, Fei Li, and Zhenbo Li. "CMFTNet: Multiple fish tracking based on counterpoised JointNet". In: *Computers and Electronics in Agriculture* 198 (2022), p. 107018. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2022.107018. URL: https://www.sciencedirect.com/science/article/pii/S0168169922003350.

[5] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online Object Tracking: A Benchmark". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.

[6] Yucheng Zhang et al. "Recent advances of single-object tracking methods: A brief survey". In: *Neurocomputing* 455 (2021), pp. 1–11. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.05.011. URL: https://www.sciencedirect.com/science/article/pii/S0925231221007220.

[7] Wenhan Luo et al. "Multiple object tracking: A literature review". In: *Artificial Intelligence* 293 (2021), p. 103448. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2020.103448. URL: https://www.sciencedirect.com/science/article/pii/S0004370220301958.

[8] Rui Yao et al. "Video Object Segmentation and Tracking: A Survey". In: *ACM Trans. Intell. Syst. Technol.* 11.4 (May 2020). ISSN: 2157-6904. DOI: 10.1145/3391743. URL: https://doi.org/10.1145/3391743.

[9] Alireza Asvadi et al. “3D object tracking using RGB and LIDAR data”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 1255–1260. DOI: 10.1109/ITSC.2016.7795718.

[10] Young-Chul Yoon et al. “Online multiple pedestrians tracking using deep temporal appearance matching association”. In: *Information Sciences* 561 (2021), pp. 326–351. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.10.002. URL: https://www.sciencedirect.com/science/article/pii/S0020025520309890.

[11] Xiaojing Li et al. “Real-Time Underwater Fish Tracking Based on Adaptive Multi-Appearance Model”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 2710–2714. DOI: 10.1109/ICIP.2018.8451469.

[12] Erik Bochinski, Volker Eiselein, and Thomas Sikora. “High-Speed tracking-by-detection without using image information”. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078516.

[13] Zhengxia Zou et al. “Object Detection in 20 Years: A Survey”. In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.

[14] Jonas Jäger et al. “Visual fish tracking: Combining a two-stage graph approach with CNN-features”. In: *OCEANS 2017 - Aberdeen*. 2017, pp. 1–6. DOI: 10.1109/OCEANSE.2017.8084691.

[15] Zhongdao Wang et al. “Towards Real-Time Multi-Object Tracking”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 107–122. ISBN: 978-3-030-58621-8.

[16] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[17] Bundesanstalt für Gewässerkunde / Federal Institute of Hydrology. *Supplementary dataset*. [usage, copying, sharing, or publication of the data without consultation and approval from the BfG is prohibited]. Accessed: 24-04-2023. URL: https://www.etis.ee/Portal/Projects/Display/669139e4-524b-4562-9aad-a1164b870823.

[18] Shilpi Gupta et al. “DFTNet: Deep Fish Tracker With Attention Mechanism in Unconstrained Marine Environments”. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–13. DOI: 10.1109/TIM.2021.3109731.

[19] Concetto Spampinato et al. “Covariance based Fish Tracking in Real-life Underwater Environment.” In: *VISAPP (2)*. 2012, pp. 409–414.

[20] Zhi-Ming Qian, Xi En Cheng, and Yan Qiu Chen. “Automatically detect and track multiple fish swimming in shallow water with frequent occlusion”. In: *PloS one* 9.9 (2014), e106506.

[21] Zhiping Xu and Xi En Cheng. “Zebrafish tracking using convolutional neural networks”. In: *Scientific reports* 7.1 (2017), p. 42815.

[22] Joao F. Henriques et al. “High-Speed Tracking with Kernelized Correlation Filters”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (Mar. 2015), pp. 583–596. DOI: 10.1109/tpami.2014.2345390. URL: https://doi.org/10.1109%2Ftpami.2014.2345390.

[23] Jane Bromley et al. “Signature verification using a"Siamese" time delay neural network”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.4 (1993), pp. 669–688. URL: http://oro.open.ac.uk/35662/.

[24] Gregory R. Koch. “Siamese Neural Networks for One-Shot Image Recognition”. In: 2015.

[25] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[26] Robert B Fisher et al. *Fish4Knowledge: collecting and analyzing massive coral reef fish video data*. Vol. 104. Springer, 2016.

[27] Meng-Che Chuang et al. “Underwater fish tracking for moving cameras based on deformable multiple kernels”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.9 (2016), pp. 2467–2477.

[28] Jerome Berclaz et al. “Multiple object tracking using k-shortest paths optimization”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819.

[29] Oliver Mothes and Joachim Denzler. “Anatomical Landmark Tracking by One-shot Learned Priors for Augmented Active Appearance Models.” In: *VISIGRAPP (6: VISAPP)*. 2017, pp. 246–254.

[30] Mygel Andrei M. Martija and Prospero C. Naval. “SynDHN: Multi-Object Fish Tracker Trained on Synthetic Underwater Videos”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 8841–8848. DOI: 10.1109/ICPR48806.2021.9412291.

[31] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[32] Yihong Xu et al. "How to train your deep multi-object tracker". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6787–6796.

[33] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. "Tracking without bells and whistles". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 941–951.

[34] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[35] Christopher R Conrady et al. "Automated detection and classification of southern African Roman seabream using mask R-CNN". In: *Ecological Informatics* 69 (2022), p. 101593.

[36] Ramil Lumauag and Marevic Nava. "Fish Tracking and Counting using Image Processing". In: *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*. 2018, pp. 1–4. DOI: 10.1109/HNICEM.2018.8666369.

[37] Hamid Hassanpour. *Euclidean Distance Filter for Image Processing*. Feb. 2021. DOI: 10.36227/techrxiv.13664615.

[38] Deepak Kumar Rout et al. "Walsh–Hadamard-Kernel-Based Features in Particle Filter Framework for Underwater Object Tracking". In: *IEEE Transactions on Industrial Informatics* 16.9 (2020), pp. 5712–5722. DOI: 10.1109/TII.2019.2937902.

[39] . *Home - Underwaterchangedetection*. http://underwaterchangedetection.eu/index.html.

[40] *ReefVid: Free Reef Video Clip Database*. http://www.reefvid.org/index.php.

[41] Wenwei Xu and Shari Matzner. "Underwater Fish Detection Using Deep Learning for Water Power Applications". In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2018, pp. 313–318. DOI: 10.1109/CSCI46756.2018.00067.

[42] Ahsan Jalal et al. "Fish detection and species classification in underwater environments using deep learning with temporal information". In: *Ecological Informatics* 57 (2020), p. 101088.

[43] Xiu Li et al. "Fast accurate fish detection and recognition of underwater images with Fast R-CNN". In: *OCEANS 2015 - MTS/IEEE Washington*. 2015, pp. 1–5. DOI: 10.23919/OCEANS.2015.7404464.

[44] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[45] Zhenxi Zhao et al. "Composited FishNet: Fish Detection and Species Recognition From Low-Quality Underwater Videos". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 4719–4734. DOI: 10.1109/TIP.2021.3074738.

[46] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[47] Ahmad Salman et al. "Real-time fish detection in complex backgrounds using probabilistic background modelling". In: *Ecological Informatics* 51 (2019), pp. 44–51.

[48] Chris Stauffer and W Eric L Grimson. "Adaptive background mixture models for real-time tracking". In: *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*. Vol. 2. IEEE. 1999, pp. 246–252.

[49] Zahra Soleimanitaleb and Mohammad Ali Keyvanrad. "Single Object Tracking: A Survey of Methods, Datasets, and Evaluation Metrics". In: *ArXiv* abs/2201.13066 (2022).

[50] Yingkun Xu et al. "Deep learning for multiple object tracking: a survey". In: *IET Computer Vision* 13.4 (2019), pp. 355–368. DOI: https://doi.org/10.1049/iet-cvi.2018.5598. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cvi.2018.5598. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cvi.2018.5598.

[51] Peize Sun et al. *DanceTrack: Multi-Object Tracking in Uniform Appearance and Diverse Motion*. 2021. DOI: 10.48550/ARXIV.2111.14690. URL: https://arxiv.org/abs/2111.14690.

[52] Zhongdao Wang et al. *Towards Real-Time Multi-Object Tracking*. 2019. DOI: 10.48550/ARXIV.1909.12605. URL: https://arxiv.org/abs/1909.12605.

[53] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric". In: *CoRR* abs/1703.07402 (2017). arXiv: 1703.07402. URL: http://arxiv.org/abs/1703.07402.

[54] Yunhao Du et al. *StrongSORT: Make DeepSORT Great Again*. 2022. DOI: 10.48550/ARXIV.2202.13514. URL: https://arxiv.org/abs/2202.13514.

[55] Yifu Zhang et al. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2021. DOI: 10.48550/ARXIV.2110.06864. URL: https://arxiv.org/abs/2110.06864.

[56] Jinkun Cao et al. *Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking*. 2023. arXiv: 2203.14360 [cs.CV].

[57] JEFFREY A TUHTAN et al. “SMART FISH COUNTER FOR MONITORING SPECIES, SIZE, MIGRATION BEHAVIOUR AND ENVIRONMENTAL CONDITIONS”. In: ().

[58] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV].

[59] Kaiyang Zhou et al. *Omni-Scale Feature Learning for Person Re-Identification*. 2019. arXiv: 1905.00953 [cs.CV].

[60] Kaiyang Zhou and Tao Xiang. “Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch”. In: *arXiv preprint arXiv:1910.10093* (2019).

[61] Liang Zheng et al. “Scalable Person Re-identification: A Benchmark”. In: *Computer Vision, IEEE International Conference on*. 2015.

[62] Jack Valmadre et al. *Local Metrics for Multi-Object Tracking*. 2021. arXiv: 2104.02631 [cs.CV].

[63] Jonathon Luiten et al. “Hota: A higher order metric for evaluating multi-object tracking”. In: *International journal of computer vision* 129 (2021), pp. 548–578.

[64] Keni Bernardin and Rainer Stiefelhagen. “Evaluating multiple object tracking performance: the clear mot metrics”. In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[65] L. Leal-Taixé et al. “MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking”. In: *arXiv:1504.01942 [cs]* (Apr. 2015). arXiv: 1504.01942. URL: http://arxiv.org/abs/1504.01942.

[66] A. Milan et al. “MOT16: A Benchmark for Multi-Object Tracking”. In: *arXiv:1603.00831 [cs]* (Mar. 2016). arXiv: 1603.00831. URL: http://arxiv.org/abs/1603.00831.

[67] *Cvat*. URL: https://www.cvat.ai/.

[68] Shalabh Aggarwal. *Flask framework cookbook*. Packt Publishing Ltd, 2014.

[69] Mark Masse. *REST API design rulebook: designing consistent RESTful web service interfaces*. " O’Reilly Media, Inc.", 2011.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.