Tracking animals in underwater videos

João Teixeira
joao.santos.teixeira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

December 2020

Abstract

Object tracking using video is an important tool that has applications in many different areas. In this work, we focus on tracking fishes, using videos recorded in the tanks of Oceanário de Lisboa, an aquarium that recreates real-life underwater conditions of different habitats. Having a reliable tracking system can help automate the monitoring of the tanks and allows the development of more complex systems like anomalous behaviour detection. We present a system that detects and tracks fish of different species in two tanks with different characteristics. There are many challenges that complicate this task like the frequent occlusion of the targets, the presence of schools or high visual similarities between different individuals. Several state-of-the-art detectors were tested to assess their performance in the two different tanks and a tracker based on color and position was developed to associate detections to existing tracks. We evaluate different variants of the system and optimize the parameters for each environment. New datasets were manually built using one video for each environment.
Keywords: Object tracking, Object detection, Fish, Video

1. Introduction

In order to study the behaviour of known fish species, it is important to observe them through long periods of time to identify patterns and reoccurring actions. Some of the behaviours that the fish exhibit may happen more regularly than others and vary between different species. How fish feed themselves, how they reproduce or which mechanisms they use to avoid their predators are all important behaviours that biologists are interested in studying to better comprehend the aquatic wildlife ecosystem. There are some difficulties in observing these animals given that they live underwater, and sometimes at very low depths, which limits the time that specialists can spend in direct contact with them.

Another very important topic related to marine wildlife is species extinction risk and endangerment. It is important to preserve all species and their habitats to avoid reaching a point where species disappear forever, particularly if the cause of extinction is human. If we combine extinction with big decreases in population of marine species we get a phenomenon called marine defaunation.

In this work we focus on video based tracking of animals captive in tanks that try to replicate the maritime environment. The recorded videos are automatically processed and then computer vision techniques are used to locate and track the fish. This tracking approach has the benefit of being able to track every fish that swims in the view field of the cameras, and is less invasive to the fish as they can move freely without any attachments. Oceanário de Lisboa has a public aquarium with more than 30 tanks and has more than 500 species of animals and plants from different habitats. It is important that the biologists continuously observe and monitor the tanks to make sure all animals are well and safe, and not showing anomalous behaviour. But having that many tanks makes it difficult to keep track of everything. Having a system that can automatically keep track of the fish could be an important tool in the observation task. This would make the work of specialists more efficient, since they would not need to spend as much time observing the tanks.

Problem Scope and Challenges. They two environments in which the videos were recorded have different characteristics. The first environment is the main tank, which consists in a large open space with different types of fish, from medium sized to big ones, including sharks, tunas and rays, among others. Given the large size of the tank, both in depth, width and length, it is difficult to capture everything, and also the distance which the camera is able to capture is relatively short meaning that fish swimming away from the camera will get lost. The fish present in this tank share a lot of similarities in their colors, being mainly gray, and their movement is also slow and without abrupt

changes in direction. There is also the presence of very big schools swimming around that can make the tracking process difficult. The coral reef tank is a lot smaller and the distance between the glass and the back of the tank is very short meaning the fish do not disappear in the distance. The fish in this tank are small in size and their movement patterns are less predictable as they change directions easily. They can also appear to be very quick given that they are closer to the camera due to the smaller size of this tank. There is a lot more color present in this environment, both in the coral reefs and in the fish.

As already mentioned, these specific environments present different challenges to tracking. First, there is the problem of occlusion, that happens when fish cross paths and momentarily hide behind other fish/objects. Then, there is the presence of schools which is heavily linked with the occlusion problem but can bring other problems in detecting other fish, as we will see later. Having a big group of fish swimming very close to each other is one of the main obstacles for the main tank. Next, we have the movement of the coral reefs with the currents, that could be mistaken as fish moving around. There are other generic problems of tracking like target rotations and scale variations or movement unpredictability.

We want to develop a system that tracks fish inside tanks. The target fish is manually identified in a frame and the system must track the fish in the following frames until it swims out of the field of view. The tracking system has to be robust in order to track fish of different sizes, colors, shapes and also has to be able to handle different tank conditions. Additionally, some of the challenges like occlusion, light variations, fish swimming at different speed and acceleration, changes in apparent size of the target have to be handled in order to adequately track the chosen target. In this work, we assume that each tank has a single camera.

Contributions. We develop a simple fish tracking system using video and computer vision techniques, that can work with multiple species of fish and in different types of habitats. First, it can equalize the colors of the underwater frames, if necessary. Next, it is able to detect fish using two different methods that can be used alternatively, one using background subtraction and the other using bounding box prediction. Then, it tracks the fish through data association, by matching each detected object with an existing track after comparing their similarities in position and color distribution. The tracking mechanism uses three tracking features that give more robustness to the tracker: color history, temporary tracks and movement prediction. A simplified system architecture of the proposed system is shown in Figure 1.

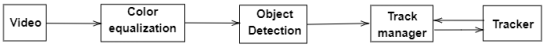

Figure 1: Simplified overview of the proposed system architecture.

We also present a comparative study on multiple state-of-the-art object detection approaches to understand which one works better in the two considered underwater scenarios. It includes traditional background subtraction approaches as well as one using deep learning.

Finally, two small datasets were manually created, one regarding fish detection and the other regarding fish trajectories.

1.1. Outline

In Section 2, we review the state of the art approaches relevant to our problem. Then, in Section 3, we present the implementation of the system developed, that details all modules of the pipeline. In Section 4, we explain the evaluation procedures, the metrics and the datasets used in each evaluation. Next, in Section 5, we present the results and then finally in Section 6, we make a summary of our work and suggest additional steps that could be implemented in future work.

2. Related Work

In this section, a revision of the state-of-the-art will be made, divided into three categories: image preprocessing, object detection and tracking.

Pre-processing techniques might be important to use in some scenarios where the colors may change due to specific conditions of the working environment. This happens because of the way water absorbs light. Colors with bigger wavelengths, like red, are absorbed at shorter depths than colors with shorter wavelengths, like blue.

Through histogram equalization we can improve the contrast of the images. There are some works on color equalization for underwater images. In [17] is presented a video based system to detect fish. The authors of that work transformed the color space and apply a histogram equalization technique called Contrast Limited Adaptive Histogram Equalization (CLAHE). This approach was used in a similar context to ours, in underwater recordings, although in their videos the water was more blurred. Another approach explained in [14], named UCM, also tries to enhance the quality of underwater images. They first make some corrections while using RGB and then again in HSI.

There are two **object detection** approaches that we considered relevant: either using background subtraction (using classical methods or deep learning) or object detectors based on a neural network. So, in this section we will review some works in

those three categories.

There are many background subtraction algorithms using classical methods, like GMM [10] [33] [37] that tries to model the background using a mixture of Gaussians and then comparing the new pixel colors with the model created to see if it is a close representation. KNN [9] [38] instead uses density estimation to model the background. Sigma-Delta [21] uses two Σ-Δ filters providing information about how motion likelihood and color variance. Then, based on these values classifies each pixel as foreground/background. ViBE [2] stores for each pixel multiple colors seen previously, know as background samples, and uses those values to compare with new pixel colors. There are update mechanisms that can propagate updates to the pixel neighbours. PBAS [13], Lobster [30], PAWCS [31] and SubSENSE [32] used the same concepts as ViBE and added more features to make it more dynamic like using automatic thresholds adjustments. GSOC is another approach based on background samples but has no paper associated to it and is included in the OpenCV contrib package.

There are also some works on background subtraction using deep learning. More recently, the study of Convolutional Neural Networks (CNN) applied to background subtraction has gained relevance. Instead of the low-level or hand-crafted features used so far, like the values of the color or the binary similarity patterns, this technique learns spatial features automatically through training. One of the first approaches was [6]. The algorithm is separated into several parts. It is a scene-specific approach meaning it has to be trained for each new scenario. A background image estimation is made using the first 150 frames by applying a temporal median over each pixel. Then, a scene-specific dataset is generated using pairs of input and background patches and calculating their target values through other background subtraction algorithms or manual labelling. The authors of [1] also propose an algorithm using CNNs that can be applied to different scenarios without retraining. They extract background samples using SubSENSE algorithm [32] to make an array of colors corresponding to the background for each pixel. This array is fixed in length and old samples are replaced by the more recent ones. To build the background image, an average of the values is computed. Then the network is trained using patches of the frames and the estimated background. FgSegNet neural network presented in [20] uses an encoder-decoder approach to segment the foreground. The encoder outputs feature maps F that are then fed into a feature pooling module (FPM) before going into the decoder. In the FPM, the features F go through multiple dilated convolutions with increasing dilation rates, where the resulting features of each convolution are concatenated with the original features F before applying the next convolution. A new feature map F' is obtained by concatenating all the resulting multi-scale features. At the end, F' is used as input to the decoder that outputs a probability for each pixel, which is then thresholded to obtain the foreground mask. One interesting aspect of this approach is that the authors were able to train the network with a small number of frames.

Besides background subtraction techniques, there are also deep learning based detectors that can locate objects and classify them. YoloV3 [27] is a CNN that is trained to predict object classes and bounding boxes in an image. It is an improved version from the original YoloV1 network [25] and also YoloV2 [26]. The way YoloV1 approaches object detection and classification by dividing the image in a $S \times S$ grid. Each cell is responsible for detecting the objects whose centers fall inside them. YoloV2 [26] was developed to improve the original approach, regarding localization errors and recall. The changes include the use of batch normalization, a higher resolution classification training for the last epochs, multi-scale training and a different model network called Darknet-19 with 19 convolutional layers. Another important difference from the previous approach is the use of anchor boxes (or priors) to predict the bounding boxes, inspired by the anchor boxes in [28]. The Yolov3 allows for multi-label predictions for each box as some labels are not mutually exclusive and also makes predictions at three different scales, which means there are priors for each scale. This time, three priors were used per cell in a total of nine across all scales. Additionally, a new model (Darknet-53) is used.

Tracking algorithms can be split into two main groups: single target trackers and multiple object trackers. One approach to track a single target is by using correlation filters like the Kernelized Correlation Filter (KCF) proposed in [12]. Using the properties of the Fourier Domain, circulant matrices and Gaussian kernels, it is possible to compute a very fast correlation filter to identify the target in the following frame. This is done by training a target regression using the sample from the current frame and in the following frame trying to find the location that better matches the filter created. Once the area with the maximum response is found, a new training is done in order to keep the target model updated. To deal with the scale variations, a scale-adaptive KCF was introduced in [19]. A scaling pool $S = \{t_1, ..., t_k\}$ with different scale variations is used and multiple patches of different

sizes of the current frame are extracted according to each element in S.

One typical approach in multiple target tracking is through data association using methods like the Hungarian Algorithm [18]. Considering that we have a set T of tracks from the previous frame, and a set d of detected objects using some kind of object detector, one can try to associate each track T_i to a detected object d_j. To do this, we have to use a similarity measure to compare each object from the current frame to the ones from the existing track set of the previous frame. After that, the Hungarian algorithm is used to maximize the similarity value for each association between the tracks and the objects. One possible feature to use in the similarity metric is based on the position of the target, by using its last known position or alternatively an estimation of the position of where the target is believed to be according to past movement. This is the approach used in [3]. After the objects are detected, a Kalman filter [15] is applied to predict the position of the previous targets in the current frame. Kalman filter is an estimator that is commonly used to predict future positions of the target in a linear movement. For each object, the position and velocity (vertically and horizontally) are used to make a prediction and then, after associating the tracks with the new objects the measured position is used to update the model. For each existing target, a bounding box is computed using the predicted position, and Intersection-over-Union is computed as a metric of similarity between the bounding boxes of the detected objects and the bounding boxes of the current targets to afterwards apply the Hungarian algorithm. In [36], an Interactive Multiple Model (IMM) is used which combines more than one model to cover different types of trajectories and conditions. They include a constant velocity model (CV) for the cases of non-maneuvering motion, constant acceleration model (CA) and constant turn model (CT) for maneuvering motion.

In [8], the problem of tracking big crowds is approached. For the targets being tracked, several candidate locations are sampled and the best locations are chosen through an objective function. The first part of the function corresponds to the appearance of the target. Next comes the target motion where the authors use a Kalman filter model to predict the targets locations. The third component tries to model the motion of the neighbouring targets, clustering the targets into different groups The fourth component is a spatial proximity soft constraint which tries to discourage the tracker from selecting locations that are too close. The last part is a grouping constraint where groups of people with similar movement are formed.

3. Implementation

We want the system to track specific fish, so the first procedure is the manual initialization of the tracks for the fish that the user wants to track, allowing selection of multiple targets. The initialization is made by selecting the bounding boxes around the targets, which are saved together with the corresponding frame numbers.

Our approach consists of three main components working in a pipeline: color equalization, object detection and tracking. A frame F^t is retrieved from a video at time t, and first goes through the color equalization module to possibly be altered in an attempt to improve the colors displayed. This module outputs the processed frame F'^t that is used in the next module of object detection. In the object detection step, we try to identify where the objects are present in the frame. The objects are detected either through background subtraction or deep learning. With background subtraction, we are able to retrieve both the mask (pixels inside the shape corresponding to the object) and the bounding box for the objects, while with deep learning, we can only get the bounding boxes. Either way, this module outputs a list of detected objects to be used later. The final module is related to the tracking step. Here, we use the list of detected objects as well as a list of currently tracked objects and try to pair a detected object with a track. Since both the detections and the tracks may not always get a match with one another, we need a fourth component, the track manager, that is responsible for the creation and deletion of tracks. The track manager also updates the tracks in case of a successful match between an object and a track.

In the following sections, each component is described in more detail.

3.1. Color equalization

This module is optional and depends on the environment displayed in the video. The colors captured in the coral reef tank video are clear enough as everything in that tank is closer to the camera and is well illuminated, so no color equalization is needed for that video. The idea behind this module was to improve the color displayed by the video of the main tank so it would be easier to later identify the fish. As this environment is not close to the surface, the lighting is poor and makes it harder to detected some of the fish that are far away from the camera as well as the ones swimming in areas where their colors are similar to those of the background. The type of transformation chosen to try to improve the frames was the one reviewed in the related work using CLAHE [17]. In that work, the authors also tried to improve the contrast of underwater images and it seemed suitable to apply in the video sequences of our main tank.

We convert the frame from its original color space RGB to CIELAB. Next, we apply the CLAHE operation included in the OpenCV library in the L channel, that represents the lightness of the color. Following the work of [17], we used a 16×16 grid with a clip limit of 2. After the transformation, we convert the frame back to RGB to be used in the following modules.

3.2. Object Detection

This module is responsible for identifying and locating fish in a given frame. We included two different types of approaches in the detection process that can be used alternatively. The first one is through background subtraction where the objects are obtained using a segmentation of the predicted foreground pixels. Each pixel is classified individually and then pixels that are close together form a blob that is considered to be a fish detection. There are many background subtraction algorithms that could be used, so a comparative study was performed on frames of our videos using different algorithms. This study, that is presented in the sections ahead, showed that the best performing algorithm for our environments was the GSOC, therefore it was the one chosen to be used in our system. The alternative to background subtraction is an object detector using a CNN that is trained to locate the fish. We use the YoloV3 net that is one of the best object detectors currently and has showed good results in other domains.

The implementation of **GSOC** is available in OpenCV contrib repository[1], and it has no paper associated to it. It uses a background model that is built and updated over time which is then compared with new frames from the video to decide for each pixel if it is in the background or in the foreground. In this particular approach, the model is based on color samples, meaning that for each pixel, a buffer of color samples is stored. The sample of this model has three important elements: the color intensity for three channels of RGB, a reference to the last frame this color was seen on in that pixel location and how many times it was seen in the past, also referred to as a hit.

The first step of GSOC is to initialize the background model. The initialization takes the first frame and creates, for each pixel position, multiple samples using the color of the corresponding pixel in that first frame and both the reference to last frame and the number of hits are set to zero. The number of background samples stored can be defined by the user.

Once the model is initialized, we can start the pixel classification process. First, we compare each pixel color with the colors from each background sample to determine what is the closest one. This comparison is made using the squared Euclidean Distance, that computes the distance D between two colors $c1$ and $c2$ as

$$D(c1, c2) = (c1_R - c2_R)^2 + (c1_G - c2_G)^2 + (c1_B - c2_B)^2 \quad (1)$$

Then, we compute the color threshold c_{thr} to decide if it is background or foreground that is given by

$$c_{thr} = \alpha * M^t + \beta \quad (2)$$

where α and β are given by the author as 0.01 and 0.0022, respectively, and M^t, which is used as an automatic manager of the threshold value and is initialized at 0.005, is given by

$$M^t = \delta M^{t-1} + (1 - \delta) D^t \quad (3)$$

using δ equal to 0.1. If the minimum distance between the new pixel color and the background samples stored for that position is higher than c_{thr}, the pixel is classified as foreground. But even if it is classified as foreground, there is still a chance that this color is used to create a new background sample, with probability $P_{replace}$ defined by the user. When a new sample is created, it replaces the oldest one (the one that has not been seen the longest). If the minimum distance is lower than c_{thr}, that position is classified as background. In that case, the color of the sample that was the most similar to the new pixel value is updated (average between the color stored and the new color) as well as the timestamp of the last hit and the number of hits. On top of that, there is also the possibility that this sample that got a match with the new color is going to replace one of the samples from the neighbours sample set. For this to happen, there is a probability given by $P_{propagation}$ and the number of hits of the sample has to be higher than the hits threshold h_{thr}.

The last step of the algorithm is to remove the noise in the segmentation. A connected components algorithm is applied to label the pixels into groups, where pixels from the segmentation mask that are connected get the same label. Then we change the classification of the very small areas. Then, a Gaussian Blur is applied to the final segmentation mask and the values are thresholded for final classification. After getting a final segmentation mask with each pixel classified, we apply again the connected components algorithm to extract a list of foreground objects that is used as fish detections d for the tracking part.

The use of an object detector like **YoloV3** results in a different working flow and a different type of output. Instead of having a background model that is initialized and then continuously updated as the frames are processed, we have a network that is

[1] https://github.com/opencv/opencv_contrib

trained beforehand using ground truth examples of real targets (fish in our case). The network used in this work had already been trained to detect a single class of objects: fish. There is no distinction between different species or groups/families. More information on the network and training can be found in Section 4. Once the training is done, we can start running the frames through the network and getting its predictions. As it was a bit slow and would take too much time to evaluate the experiments, the bounding box predictions for every frame are made previously and saved. Then for each frame, a list is created with the bounding boxes and corresponding areas of the frame. This is where the output can have some differences between approaches: using GSOC the module outputs a list of blobs that can have multiple shapes and using Yolo the outputs are always rectangular.

3.3. Tracking

Our approach is a multi-target approach instead of a single-target one, as we create tracks for every new fish detected in the video instead of only creating a track for the selected fish. Having tracks for every fish can help us prevent incorrect associations, as we have to maximize the association of all tracks.

The first important element in this module is the track. A track T_k corresponds to a trajectory of a detected fish. In this case, a trajectory is the temporally ordered list of frame coordinates of where the fish has been detected and the corresponding frame numbers of when those detections occurred. We want to allow tracking fails for a certain amount of frames before terminating the track instead of terminating as soon as the tracking fails, as we need to account for the detection fails. So, each track will not have a corresponding detection in every frame and will have some jumps along the trajectory. In addition to the position and frame number, there are other elements stored that are important to tracking. Each track has a different ID so we can tell them apart and understand if the associations are being successful and consistent over time. We also store an image of the target and use it to build an RGB histogram that is used in the association step. Then we have a variable indicating if the track was initialized by the user or not so we can end the tracking when there are no more active tracks $\hat{T}$ selected by the user. An active track has an indicator to show that it is still being used for the tracking. There is also a counter for the number of consecutive frames in which the track did not match with any detected object, that will be used to turn active tracks into inactive ones.

The next component is the track manager. It is responsible for adding new tracks and update them over time as well. When there is a new detection that was not associated in an existing track, a new track has to be created by the track manager. Every time there is a new frame to be tracked, the track manager looks up in the list of all tracks, the ones that are still active. After the association between detected objects and existing tracks, the update step occurs. Every track with a successful match is updated to include the new bounding box, build a new histogram for the new image and set the counter of consecutive fails to 0. For every detection that that did not match with an existing track, a new track is initialized. For every existing track without a match, the consecutive fails counter is increased. If the counter becomes higher than the maximum allowed (set to 48 frames), the track becomes inactive.

Once we have the list of detected objects and the list of active tracks $\hat{T}$, we can progress on the tracking procedure. To perform the data association we need to compute the similarities between each element of one set with the elements of the other, obtaining a similarity matrix. There are two components used to compute the similarity S between a active track $\hat{T}_k$ and a detection d_j, color and position, and is given by

$$S(\hat{T}_k, d_j) = S_c(\hat{T}_k, d_j) + S_p(\hat{T}_k, d_j) \tag{4}$$

where S_c is the color similarity and S_p is the position similarity. To compute S_c, we use the track's RGB histogram H and build one for the detection. Then we compute the Hellinger distance D_H between the two, that returns a value between 0 and 1, and is given by

$$D_H(H_{\hat{T}_k}, H_{d_j}) = \sqrt{1 - \frac{1}{\sqrt{H_{\hat{T}_k} H_{d_j}}} \sum_I H_{\hat{T}_k}(I) * H_{d_j}(I)} \tag{5}$$

and we use this value to compute S_c

$$S_c(\hat{T}_k, d_j) = 1 - D_H(H_{\hat{T}_k}, H_{d_j}) \tag{6}$$

The position similarity is computed as

$$S_p(\hat{T}_k, d_j) = 1 - \frac{\sqrt{(\hat{T}_{k_x} - d_{j_x})^2 + (\hat{T}_{k_y} - d_{j_y})^2}}{D_{max}} \tag{7}$$

where D_{max} is equal to 866, which is the approximate maximum possible Euclidean distance between two pixels in a 720×480 frame. We convert the similarity matrix to a cost matrix and apply the Hungarian algorithm, that optimizes the associations between all tracks and detections. Then, there is a final step where we verify if the associations should be allowed or not. So, for each pair of detection and track, we verify if the color or the distance is not too different, and if it is, we discard

that pair. The color similarity should be higher than the color threshold c_{thr} and the distance should be smaller than distance thresholds d_{thr}. This is what the baseline of our tracking approach consisted of. While testing the system and evaluating possible improvements based on the failures happening on the validation dataset, three new tracking features where added in order to try to solve those issues.

Color history. This feature was motivated by the fact that when fish are close together or even in front of each other, the detection mask, or bounding box, will contain features from both fish. If the other fish have different colors, the histograms will get corrupted and it may prevent the correct association later on, after they are detected separately. To solve this, instead of only using the last image to create the histogram, we compute an average of the histograms allowing us to have some kind of color history. This average using γ as the weight of the new histogram is computed according to

$$H^t_{\hat{T}_k} = (1-\gamma) * H^{t-1}_{\hat{T}_k} + \gamma * H_{d_j} \quad (8)$$

Temporary tracks. We noticed that there were times when the segmentation was fragmented, producing more than one bounding box for the same fish or additional bounding boxes were incorrectly being predicted in areas around the fish. In these cases, new tracks will be created for the extraneous detections and it can prevent correct associations. To avoid this, we added a temporary track mechanism. When a track is created for a new detection, it is initialized as a temporary track. Next, we make a first application of the Hungarian algorithm only using the permanent tracks, and then remake the association step using the unmatched detections and the temporary tracks. This way the permanent tracks will have priority over the temporary ones. Once a temporary track gets 5 successful matches, it turns into a permanent track.

Movement prediction. Sometimes fish are not detected because they go behind other fish or simply because the detection failed. So it is important that the track does not use the last known location for the position similarity, as the fish will eventually get too far away if there are many consecutive matching fails. This is where the Kalman Filter can be used in order to predict the movement of the fish based on the past positions and predictions. For each frame, the position of each tracked fish is predicted. This value is then used in the position similarity computations instead of the position of the last bounding box of the track. Once there is a successful association, the prediction is updated to include the position of the new detection. If there is no associated detection for a track in a certain frame, there is no update step to the movement prediction, but the Kalman Filter will keep predicting new positions.

Table 1: Background subtraction dataset

Environment	Description	Quantity	Usage
Main tank	Pairs of frames and segmentations	8	Testing
Coral reef tank	Pairs of frames and segmentations	8	Testing
Main tank	Pairs of augmented frames and segmentations	25	CNN training
Coral reef tank	Pairs of augmented frames and segmentations	25	CNN training

Table 2: Tracking dataset

Environment	Quantity	Shortest trajectory	Longest trajectory	Usage
Main tank	15	152	827	Validation
Coral reef tank	17	58	1687	Validation
Main tank	4	197	385	Testing
Coral reef tank	4	192	264	Testing

4. Evaluation

To test and evaluate our approach, one video for each of the environments was used. Both videos were recorded at Oceanário de Lisboa using a stationary camera positioned outside the tanks, so there is a glass wall between the camera and the water. Each environment, which were already described in Section 1, presented different challenges

Using those two videos mentioned, two different datasets were built. The first one, in Table 1, is for the detection tests and the second one,in Table 2, was built for the tracking evaluation. Both of them were manually created.

To **evaluate the object detection**, we separate the experiment in two parts: background subtraction and bounding box prediction. For the background subtraction, each variant of the test is described in Table 3.

Table 3: The different variants of the background subtraction tests using the testing dataset and in which tanks they are used.

Variant	Description	Tanks
Single-pass	The base variant where each algorithm is applied to the video once.	Both
Two-pass	First run build background model, and in second evaluate.	Both
Color equalization	Verify if color equalization helps the algorithms.	Main tank

To train the NN FgSegNetv2, we follow the same procedure as the authors [20]. To evaluate object detector YOLO we compare the outputs with manually labeled bounding boxes. To have some similarities with the previous part of the experiment, the same eight frames were considered. In these eight frames of each environment, the fish present in them were manually selected by drawing a bounding box around it but the selection varied between

environments. In the coral reef tank, all fish were selected and in the main tank, the selection was made carefully, so when comparing the results of the two tanks, one should take this into account. The network used was already trained and is available online [2] as well as the training dataset used.

Then we **evaluate the tracking** performance. First, the validation dataset will be used and the tests will allow us to tune the parameters of the tracking module as well as understand the influence of each feature. Each test will consist of one run per trajectory and the tracker will be initialized using the first bounding box of the ground-truth trajectories that were manually labeled. We will run these tests on the validation dataset using two different object detection techniques to tune the parameters. The variants of the tracking test for each video is described in Table 4. After that, we will use the testing dataset to directly compare both approaches in each environment and get our final results from there.

Table 4: The different variants of the algorithm to be tested using the validation dataset for each tannk. In each variant, the corresponding parameters are tuned.

Variant	Description
Baseline	Hungarian algorithm using color and position
1	Addition of the color history feature
2	Addition of the temporary tracks feature
3	Addition of the movement prediction feature

5. Results

In this chapter, we will present the results of the experiments described previously, obtained in a computer equipped with Intel Core i7-8750H @ 2.20 GHz CPU and GeForce GTX 1050 Ti GPU.

Background subtraction. In this experiment we started by testing multiple algorithms in both environments (main tank and coral reef tank) and measuring the time it took to process the entire videos as well as computing the F_1-Score for the selected frames. The results are presented in Figure 2 for the coral reef tank on the top and for main tank on the bottom. For the coral reef, the approach with the best F_1-Score was the FgSegNet with 0.8326 followed by PAWCS, Lobster and SubSENSE. But between all these, there is not one that can perform above 7 FPS making them very slow in comparison with the speed of our videos (24 FPS). Despite not having a requirement to run the system in real time, we still want it to be as fast as possible while being very accurate. The next best algorithm is GSOC scoring 0.6353 at 37 FPS which is a very decent performance. On the main tank, the results are a bit worse. FgSegNet is again the best algorithm with 0.6918 of F_1-Score

[2] https://github.com/rocapal/fish_detection

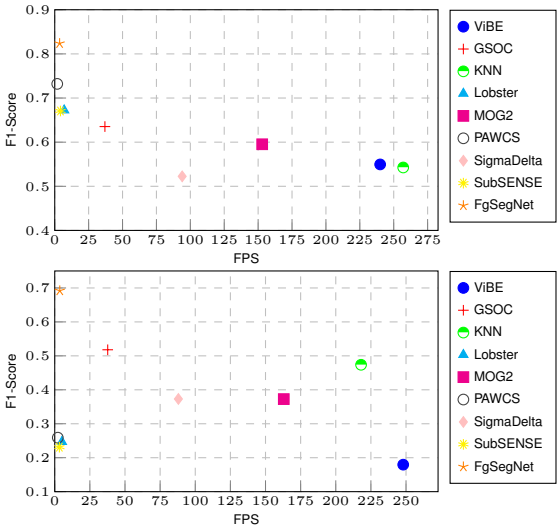

Figure 2: F1-score testing results for the algorithms in the coral reef tank on the left and the main tank on the right.

followed by GSOC with 0.5179. Every single algorithm performed better in the coral reef tank than in the main one. This was expected given that the main is at more depth causing more blur, there are schools present and the colors are more similar. This makes the object subtraction task a lot more difficult, resulting in worse results.

The results of the two-pass variant and comparison with the single-pass variant are presented in Figure 3 with the coral reef tank at the top and the main tank at the bottom. In the coral reef tank, there was no global evidence of improvement in terms of F_1-Score. Still, we were able to achieve better top performances than what we had while running only once. The PAWCS was the top scorer with 0.8220 and GSOC came in second with 0.7414. We have to take into consideration the extra time that is spent doing an extra pass through the entire video. In the main tank, there was a clear indication of improvements while running the video twice, as all algorithms benefited from it. The best algorithms were GSOC scoring 0.5354 at 18 FPS and KNN scoring 0.527 at an impressive speed of 109 FPS. The main tank is still performing worse than the coral reef tank.

The results of the last variant of this experience (color equalization on versus off for the main tank) can be seen in Figure 4. The processing speed was decreased a bit as it also includes the equalization part. Once again, the best algorithms were FgSegNet with the best score but low FPS followed by GSOC.

GSOC Tuning. Overall, GSOC was one of the best algorithms with a good trade-off between speed and F_1-Score in most of the experiment variants, for both the main tank and the coral reef one. Therefore, that was the chosen algorithm to be

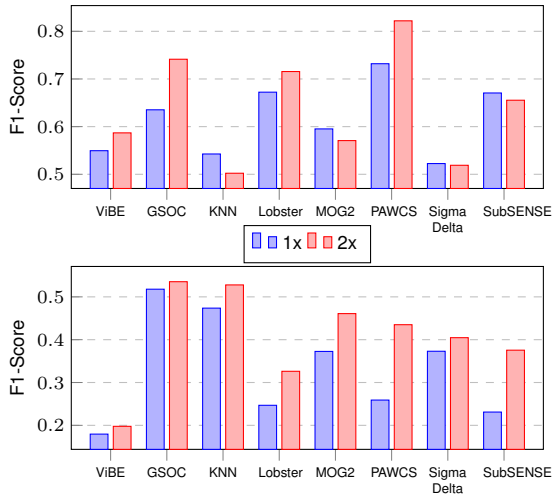

Figure 3: F1-score comparison for the single-pass and two-pass variants for the coral reef tank at the top and for the main tank at the bottom.

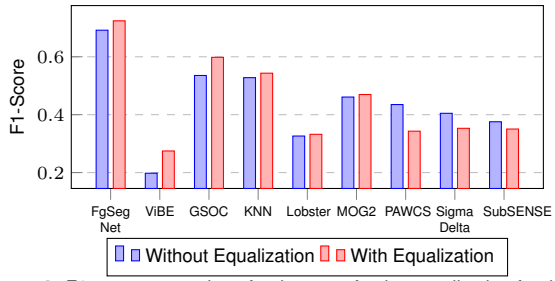

Figure 4: F1-score comparison for the use of color equalization for the main tank. The variant without equalization used with FgSegNet is the single-pass variant and the rest of them use the two-pass variant.

used and now follows the tuning of its parameters. This parameter tuning was performed on the same testing dataset that was used in the previous tests. For the coral reef tank, we use the GSOC variant that runs the video twice. For the main tank, we go with the variant that applies the color equalization and runs the video twice. We studied each parameter individually, and obtained the following configurations: the coral reef uses 80 samples per pixel, 0.005 propagation rate, 0.001 replace rate and 40 hits threshold; the main tank uses 20 samples, 0.005 propagation rate, 0.005 replace rate and 32 hits threshold. The final F_1-Score for the coral reef tank is 0.8721 and for the main tank is 0.5969.

Bounding box prediction. The results of this experiment are presented in Table 5. In the coral reef tank there are almost as many good detections as fish missed by the detector, leading to a lower Recall (0,5063). In terms of Precision (0,6896), it scored higher as there were less detections that did not correspond to a real fish than good detections. Overall, most misses corresponded to the smaller fish that were not detected and also the ones that were under the coral in an area with low light and most bad detections were consistently around the same areas of the coral reef. In the main tank there were even less bad detections resulting in a high Precision (0.8372) while Recall was again low (0.5806). In this scenario, the very few bad detections were mainly on high areas where the far away fish and the lights mixed together.

Table 5: Performance of YOLO object detector for both environments

	Main	Coral reef
TP	36	40
FP	7	18
FN	26	39

	Main	Coral reef
Precision	0,8372	0,6896
Recall	0,5806	0,5063

Now, we present the **tracking results** using two different object detection techniques, where we first tune the parameters using each one and then compare their final performance using the testing dataset.

Tracking using GSOC. The results of tracking in the coral reef tank using GSOC for the validation dataset are presented in Figure 5, on the top row. In this tank, the performance was considered to be good, with good prediction for the majority of the frames. The best configuration obtained was using color threshold c_{thr} and distance threshold d_{thr} set to 30% and 65, respectively, using color history with weight 20% for the new histogram, temporary tracks and movement prediction. In the coral reef tank tank, some of the fish move quickly sometimes so having movement prediction helped the system recover tracks after failing for some frames, as the movement kept being predicted and kept lower distance than the one allowed for an association. The temporary tracks also helped in some cases were the segmentation was fragmented for the same fish and prevented a newly created tag for one of those segments ”stealing” the correct match for the fish we were following when the segmentation was unified again.

In the main tank, the overall performance of the system is worse. The results are presented in Figure 5 on the bottom row. The best configuration obtained was using c_{thr} and d_{thr} set to 20% and 65, respectively, using color history with weight 20% for new histogram, temporary tracks and movement prediction. The addition of the last two features separately made it worse, but together the results improved. So, overall the main tank was a lot more difficult to track and one of the main reasons is the poor object detection obtained by using the GSOC background segmentation technique. Also the fish have a lot more similarities in color, with a few being different, and they also blend in with the blue background, which is one of the reasons why the segmentation performed poorly too. We were still able to track correctly some of the trajectories.

Tracking using YOLO. The results for the track-

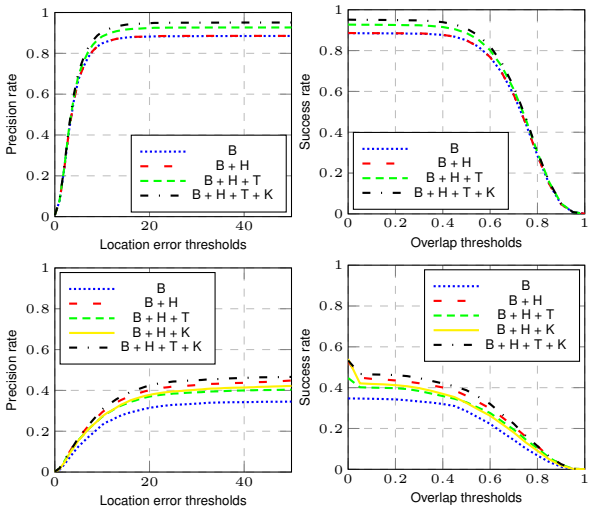

Figure 5: Precision and Success plots using GSOC for the training dataset for the different system implementations; top: coral reef tank; bottom: main tank; left: as a function of location error; right: as a function of overlap. B - baseline, H - average histogram, T - temporary tracks, K - kalman filter.

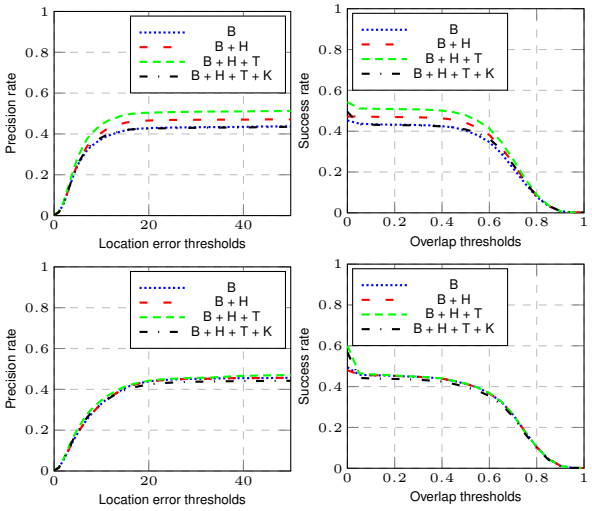

Figure 6: Precision and Success plots using YOLO for the training dataset in the coral reef tank for the different system implementations; top: coral reef tank; bottom: main tank; left: as a function of location error; right: as a function of overlap. B - baseline, H -average histogram, T - temporary tracks, K - kalman filter.

ing using YOLO for the validation dataset are presented in Figure 6. For the coral reef tank (top row of the figure), we can immediately see a big decline of the results compared to the background subtractor GSOC. The best configuration for this variant was tuning c_{thr} and d_{thr} to 20% and 65, respectively, using color history with weight 20% for new histogram, temporary tracks and without using movement prediction. Comparing the overall results in the validation dataset, it was clearly better to use the background subtractor GSOC instead of the object detector YOLO in the coral reef tank.

As for the results for the main tank (bottom row of same figure), there is a much smaller difference between using the two types of detection. The configuration for YOLO was using c_{thr} and d_{thr} set to 30% and 80, respectively, using color history with weight 90% for new histogram (even though there were not noticeable improvements), temporary tracks and without using movement prediction once again. So, like in the coral reef tank, the system performed better when using histogram averages and temporary tracks but with the movement prediction turned off, while using Yolo as the object detection technique.

Tracking: GSOC vs YOLO. This experiment uses the testing dataset so we can get the final results and make direct comparison between the use of both detectors, using the configurations from the last tests. The final results for the coral reef tank are laid out in the top row of Figure 7. We can see that GSOC achieved the best results in the tracks of this small testing dataset in both metrics. With YOLO, there was a noticeable decrease in the results but overall it was still a good performance. For all 4 tracks of this testing dataset, both versions

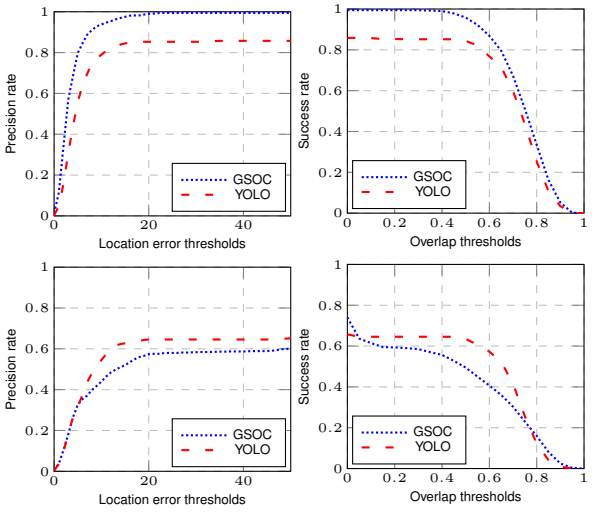

Figure 7: Precision and Success plots for the system implementations with each object detection technique for the testing dataset in the coral reef tank; top: coral reef tank; bottom: main tank; left: as a function of location error; right: as a function of overlap.

were able to track them from the start to the end, meaning that the difference in results is probably caused by more failed detections or failed associations in some frames along the way by the YOLO version than by the GSOC version. For the main tank, the results are shown in the bottom row. The results in the main tank are worse, like what happened using the validation dataset. This time the best tracking was achieved while using YOLO as object detector. These values can be explained by the loss of the targets being tracked in both system versions. Using YOLO, there were 2 tracks followed from start to end, 1 track that was lost near the end and the last one failed around the end of the first half of the track. As for GSOC, it also tracked 2 tracks from start to finish, but one track failed really early on and the other did not get to half way too.

So, overall, is better to use the background subtraction algorithm in cases like the coral reef tank, where the fish are not always together (but can still cross paths) and are smaller. As the segmentation is not dependent on the form of the fish, they can still be detected in very different shapes and orientations as long as the detection algorithm can discriminate the fish's colors from the background ones. On the other hand, in environments like the main tank, with a huge number of fish present, where they frequently swim close together, it is better to use an object detector like YOLO that is capable of identifying fish separately even if they are touching each other in the frame space. In terms of tracking features, all the best versions used histogram averaging, that was used as a history of the recent histograms, and temporary tracks, that gave less priority to tracks that were recently created. As for movement prediction, it was an improvement while using background subtraction but it decreased the performance while using the object detector YOLO.

6. Conclusions

In this work, we attempted to solve the problem of video based tracking applied to fish. There are many difficulties imposed by the natural conditions of the marine life such as multiple fish present in the same area, the camouflage behaviour that the animals use, a lot of visual similarities between different individuals of the same species and sometimes between different species, that are hard to be detected by non-specialists. A tracking system was presented that includes a color equalization module, a detection module and the tracking module. There were used two different detection approaches. The tracking is done through data association using color and position, with three added features: color history, temporary tracks and movement prediction. Overall, the results were satisfactory, but there are some limitations to our approach like the inability to handle the big fish schools swimming around as they make it very hard to track each individual fish. The detection does not work well with fish that are very far away or the very small ones in the coral reef tank so tracking these is not possible.

In the future, research can be done to improve the detection module like combining both types of detection approaches or developing a mechanism to detect fragmentation in the segmentation of the fish. For the tracking module, could be also important to study the integration of CNN features in similarity matrix computations.

References

[1] M. Babaee, D. T. Dinh, and G. Rigoll. A deep convolutional neural network for video sequence background subtraction. *Pattern Recognition*, 76:635 – 649, 2018.

[2] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, 2011.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[4] G. Bilodeau, J. Jodoin, and N. Saunier. Change detection in feature space using local binary similarity patterns. In *2013 International Conference on Computer and Robot Vision*, pages 106–112, 2013.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] M. Braham and M. Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–4, 2016.

[7] François Chollet et al. Keras. `https://keras.io`, 2015.

[8] A. Dehghan and M. Shah. Binary quadratic programing for online tracking of hundreds of people in extremely crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):568–581, 2018.

[9] Ahmed Elgammal, David Harwood, and Larry Davis. Nonparametric model for background subtraction. In David Vernon, editor, *Computer Vision — ECCV 2000*, pages 751–767, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[10] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, UAI'97, pages 175–181, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[11] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2012.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[13] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–43, 2012.

[14] K. Iqbal, M. Odetayo, A. James, R. A. Salam, and A. Z. Hj Talib. Enhancing the low quality images using unsupervised colour correction method. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 1703–1709, 2010.

[15] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[16] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground–background segmentation using codebook model. *Real-Time Imaging*, 11(3):172 – 185, 2005. Special Issue on Video Object Processing.

[17] D. Konovalov, A.Saleh, M. Bradley, M. Sankupellay, S. Marini, and M. Sheaves. Underwater fish detection with weak multi-domain supervision. *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[18] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[19] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, pages 254–265. Springer International Publishing, 2015.

[20] Long Ang Lim and Hacer Yalim Keles. Learning multi-scale features for foreground segmentation. *Pattern Analysis and Applications*, 23(3):1369–1380, 2020.

[21] Antoine Manzanera and Julien C. Richefeu. A new motion detection algorithm based on sigma-delta background estimation. *Pattern Recogn. Lett.*, 28(3):320–328, February 2007.

[22] Douglas J. McCauley, Malin L. Pinsky, Stephen R. Palumbi, James A. Estes, Francis H. Joyce, and Robert R. Warner. Marine defaunation: Animal loss in the global ocean. *Science*, 347(6219), 2015.

[23] Camilo Mora, Derek P Tittensor, Sina Adl, Alastair GB Simpson, and Boris Worm. How many species are there on earth and in the ocean? *PLoS Biol*, 9(8):e1001127, 2011.

[24] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[26] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] Andrews Sobral. BGSLibrary: An opencv c++ background subtraction library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013.

[30] P. St-Charles and G. Bilodeau. Improving background subtraction using local binary similarity patterns. In *IEEE Winter Conference on Applications of Computer Vision*, pages 509–515, 2014.

[31] P. St-Charles, G. Bilodeau, and R. Bergevin. A self-adjusting approach to change detection based on background word consensus. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 990–997, 2015.

[32] P. St-Charles, G. Bilodeau, and R. Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, 2015.

[33] C. Stauffer and W. E. 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)*, volume 2, pages 246–252, 1999.

[34] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan. Static and moving object detection using flux tensor with split gaussian models. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 420–424, 2014.

[35] M. Wu and X. Peng. Spatio-temporal context for codebook-based dynamic background subtraction. *AEU - International Journal of Electronics and Communications*, 64(8):739 – 747, 2010.

[36] R. Zhou, K. Zhou, M. Wu, and J. Teng. Improved interactive multiple models based on self-adaptive turn model for maneuvering target tracking. In *2018 Eighth International Conference on Information Science and Technology (ICIST)*, pages 450–457, 2018.

[37] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31. IEEE, 2004.

[38] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773 – 780, 2006.