Video Based Live Tracking of Fishes in Tanks

José Castelo, H. Sofia Pinto, Alexandre Bernardino, and Núria Baylina

Instituto Superior Técnico, Lisbon (INESC-ID & ISR); Oceanário de Lisboa

Abstract. We explore video tracking and classification in the context of real time marine wildlife observation. Among other applications it can help biologists by automating the process of gathering data, which is often done manually. In this paper we present a system to tackle the challenge of tracking and classifying fish in real time. We apply Background Subtraction techniques to detect the fish, followed by Feature Matching methods to track their movements over time. To deal with the shortcomings of tracking by detection we use a Kalman Filter to predict fish positions and a local search recovery method to re-identify fish tracks that are temporarily lost due to occlusions or lack of contrast. The species of tracked fish is recognized through Image Classification methods, using environment dependent features. We developed and tested our system using a custom built dataset, with several labeled image sequences of the fish tanks in the *Oceanário de Lisboa*. The impact of the proposed tracking methods are quantified and discussed. The proposed system is able to track and classify fish in real time in two scenarios, main tank and coral reef, reflecting different challenges.

Keywords: Object Detection · Video Tracking · Image Classification · Real Time · Fish

1 Introduction

Almost everything we know about animals comes from observing and analyzing their behaviours. In the past, biologists spent most of their time observing their subjects. Advances in technology allowed for better ways to make these observations. Film cameras proved to be a powerful tool when collecting data, as they convey more information than still pictures. Analog films allowed biologists to study behaviour or trait in more detail, as they can be played back multiple times. Furthermore, rare phenomenons, which could take a lifetime to encounter and understand, become objectively shareable among the community. Finally, the appearance of the digital video camera unlocked the potential to automatically process videos.

Video observations of animals, as everything, come with a set of advantages: a non intrusive way of observing animals, unaffected by the human presence; continuous observation, potentially revealing previously unseen behaviours that take place under very specific scenarios; an objective portrait of animals and their traits/behaviours, not limited by the observation of the biologists in the field; relatively low cost, as cameras and computing power become more economically

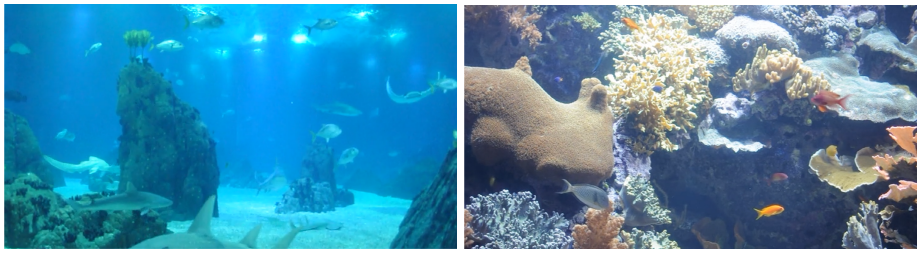


Fig. 1. Two of the tanks at the *Oceanário de Lisboa*.

viable; the possibility of remote observation or analysis. However, they have also drawbacks: produce a significant amount of data, requiring considerable amounts of data storage capacity; their output needs to be processed, as only small portions of the observations convey useful information; a limited viewport, in the sense that only what is in front of a camera is captured, potentially leaving out relevant information.

The *Oceanário de Lisboa* hosts a range of exhibition tanks that simulate environments from marine biomes all around the globe, two of which can be seen in Fig. 1. Part of its work, whether for maintenance, monitoring or research purposes, involves close observation of fish. We captured image sequences from some of its main tanks to create a dataset that captures significant diversity of scenarios and allow us to develop and test a system to automatically track and identify several classes of fish.

The *Oceanário de Lisboa* is interested in detecting many events that require a timely and long term observation of the fish, for example, when a given species is displaying a behaviour that is rarely seen. This motivated us to enable our system to constantly monitor the observed tanks and relay information regarding fish species and trajectories in real-time. Different environments and tanks, as illustrated in Fig. 1, have distinct properties and challenges: (1) the depth of each tank influences the perceived colors typically resulting in a blue tint; (2) the size of the fish greatly impacts their behaviour, with smaller fish displaying faster and more unpredictable movements; (3) the uniformity of certain groups of fish species makes their distinction in videos an extremely difficult task; (4) the size of certain tanks limit the visibility of fishes that are swimming at considerable distances from the camera; (5) the presence of marine flora hinders detecting fish, due to their movement and provided camouflage (6) variations in lighting present a constantly changing environment; (7) tank features, such as rocks, conceal the fish swimming behind them.

The Fish4Knowledge project ([1] [8] [6] [5]) aimed at surveying fish populations with a remote network of underwater cameras. They successfully applied computer vision techniques to the captured footage in order to track and classify the fish. Our scenarios pose different challenges to the ones seen in Fish4Knowledge, in particular the much shorter ranges and lower resolution footage.

2 Problem Statement and Objectives

The problem of tracking and classifying fish in real-time can be broken into three major sub-problems as follows:

1. Detect the set of fish present in each video frame t, $D^{(t)} = \{d_{n_t}^{(t)}\}, n_t = 1..N_t$, where N_t is the number of fish at frame t and $d = (x, y, w, h, m, t)$ where x, y are the horizontal and vertical bounding box position coordinates, w, h its width and height, respectively, m a mask of the pixels within the bounding box and t the frame timestamp.
2. Associate each detected fish, $d_{n_t}^{(t)}$ with the appropriate track $T_k = \{d_{i_k(t)}\}, t = t_{0k} : t_{fk}$, where $i_k(t)$ is the index of the detected blob at timestamp t which corresponds to track T_k.
3. Identify the species, c_i, out of the species set $C = c_0 : c_N$, corresponding to each track T_k.

A few restrictions and assumptions have to be taken into consideration, based on our environment: (1) the scene is captured by a statically positioned camera, from outside of the tanks to minimize the intrusiveness of the system; (2) the resolution of the video output by the camera must be at least 720×480 pixels. A lower resolution means there is not enough detail to analyse the scene; (3) there is a limited number of fish in each species which limits the size of the dataset used to train the fish classifiers; (4) a closed environment is assumed, meaning that the possible species of the observed fish will not change from those originally in the system.

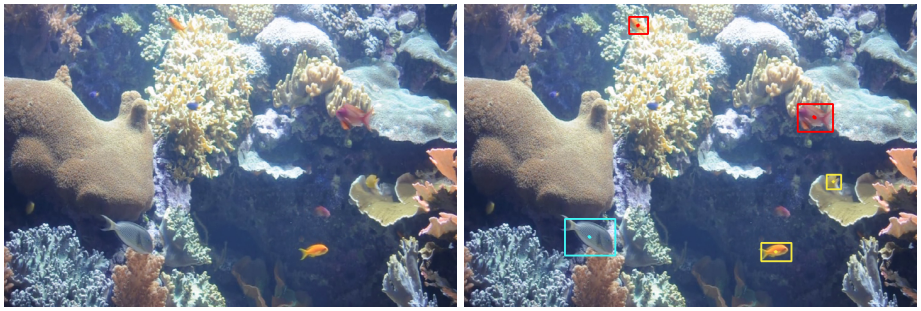
We aim at developing a practical and low cost fish observation system through the use of videos, whether for research, monitoring or entertainment purposes. By providing video tracking and species classification of fish in real-time, we seek to provide the framework for applications that require a real-time information feed containing the position and species of each fish.

Our primary objective is to provide a live feed of location and species for each fish in a video stream of a tank. An example frame of a tank video stream can be seen in Fig. 2(a), where the desired output is the location of each fish (highlighted with a bounding box) and species (bounding box color), exemplified in Fig. 2(b).

Our system should be able to support different environment conditions that fish can be observed in, as well as the ability to adapt to new environments with little setup and parameterization. It is our goal to have the system running in real-time on ordinary computers. This makes our system economically viable for most people, with no need for extra investment in specialized hardware.

3 Detection

The detection module is responsible for localizing the set of fish, $D^{(t)}$, present in video frame t. To do this, it must first be able to recognize the parts of a fish. We used the Adaptive Gaussian Mixture Model (AGMM [4]) implemented



(a) Example frame (b) Desired output

Fig. 2. Goal: live feed of locations and species for each fish in a tank.

in *opencv* that follows the specification in [9] and is composed of three stages: Background modelling, Background subtraction and Connected Components. In the following paragraphs, we succinctly describe each of the steps.

Background Modelling - In the AGMM algorithm, each pixel in the background is modeled as a Gaussian Mixture. During the background modelling stage the Gaussian mixtures for each pixel are updated. The Gaussians in the mixture are given a weight, based on their relevance to the model. This allows the model to contain information regarding more infrequent samples, while not overestimating their importance. At each update, every Gaussian has its weight, mean, and variance updated according to the newly observed data. The modelling process is detailed next.

Consider the sample history of the last T timesteps, with the rgb color values for each pixel, $\boldsymbol{x}^{(t)}$, at timestep t: $X_T = \boldsymbol{x}^{(t)}, ..., \boldsymbol{x}^{(t-T)}$. At each time step t, an estimated probability density $\hat{p}$ function models the observed values, which might contain both foreground (FG) and background (BG) related values:

$$\hat{p}(\boldsymbol{x}|X_T, BG + FG) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\hat{\boldsymbol{\mu}}_m, \hat{\sigma}_m^2 I) \tag{1}$$

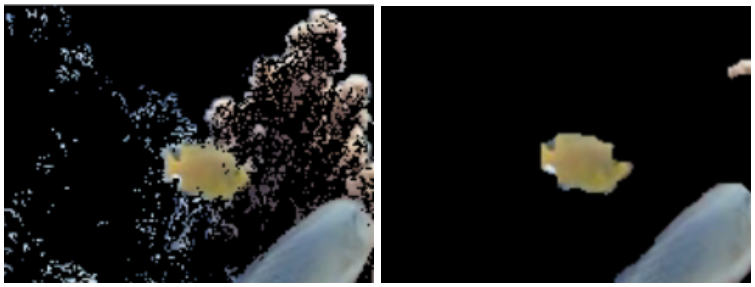
In this equation $\hat{\boldsymbol{\mu}}_m$, $\hat{\sigma}_m^2$ and $\hat{\pi}_m$ are respectively mean, variances and weight estimates for each Gaussian component m. When a new value $\boldsymbol{x}^{(t)}$ is observed, each component is updated recursively as follows:

$$\hat{\pi}_m \longleftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) \tag{2}$$

$$\hat{\mu}_m \longleftarrow \hat{\mu}_m + o_m^{(t)}(\alpha/\hat{\pi}_m)\boldsymbol{\delta}_m \tag{3}$$

$$\hat{\sigma}_m^2 \longleftarrow \hat{\sigma}_m^2 + o_m^{(t)}(\alpha/\hat{\pi}_m)(\boldsymbol{\delta}_m^T \boldsymbol{\delta}_m - \hat{\sigma}_m^2) \tag{4}$$

where $\boldsymbol{\delta}_m = \boldsymbol{x}^{(t)} - \hat{\boldsymbol{\mu}}_m$ and α is the learning rate parameter. $o_m^{(t)}$ is a binary selector, whose value is 1 when a component is the most relevant (largest weight π_m) within the components close to the observed value, according to the Mahalanobis Distance [3], and 0 otherwise. In the case where there are no close



(a) Raw foreground (b) Filtered foreground

Fig. 3. Applying morphological filters (opening and closing) to the pixels flagged as foreground.

components, a new one is generated. If the amount of components exceeds the limit, the least important (lowest weight π_m) component is discarded.

Background Subtraction - During this stage a background subtraction algorithm identifies which pixels belong to the foreground. This is done by first considering the N most important Gaussian components for each pixel. N is determined as:

$$N = argmin_n(\frac{\sum_{i=0}^{n} \hat{\pi_i}}{\sum_{i=0}^{N_m} \hat{\pi_i}} >= 0.75) \tag{5}$$

which translates into the N most important components that exceed a total summed relative weight of 75%. These components model the colors that most likely belong to the background. Then we check if the Mahalanobis Distance between the observed value $x^{(t)}$ and any of these N components is bellow a threshold (80 in our case, determined empirically). As the used implementation considers that the color channels are independent, their covariance matrix is the Identity matrix. In this case, the Mahalanobis Distance [3] between a Gaussian component m with mean value $\boldsymbol{\mu}$ and an observed value $\boldsymbol{x}$ is:

$$D_M(\boldsymbol{\mu}, \boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T} \tag{6}$$

Once all the pixels are processed we now have a frame labeled into background and foreground pixels. An example of a set of pixels that have been labeled as foreground can be seen in Fig. 3(a).

Morphological Operations and Connected Components - The foreground (Fig. 3(a)) is then manipulated with morphological filters. These help remove noise, such as isolated pixels identified as foreground, and close small holes within big regions of foreground pixels. The impact of these operations can be seen in Fig. 3(b). Each separate foreground region is then uniquely identified by a two-pass connected component algorithm. This process identifies pixels identified as foreground that form connected regions, uniquely identifying each separate region.

4 Tracking

The first important aspect is to discuss the features to take into account when comparing a blob to a track, and how this comparison is made.

Position Similarity - The position of a fish does not usually change significantly between consecutive frames. Moreover, this movement is relatively predictable given the short time elapsed between frames. As such, the position of the detected blobs are compared to the predicted position of each track. This prediction is computed through a Kalman Filter, which takes into account the position history of each track. A Kalman Filter also takes into account noise in position measurements, which could occur, for example, due to faulty detection. For a given detected blob d_j and track T_k, the position similarity function is computed as follows:

$$\mathcal{S}_p(d_j, T_k) = 1 - \frac{\|P_K(T_k) - P(d_j)\|}{M_d} \tag{7}$$

where M_d is the maximum Euclidean Distance in the frame, which can be computed based on the video resolution. $P(d_j)$ is the position of blob d_j and $P_K(T_k)$ is the position predicted by a Kalman Filter for the track T_k.

Color Similarity - The second feature is based on the appearance of a fish. It consists of a color description of a blob, computed in the form of a relative frequency color histogram. This normalized histogram contains a set of bins for each color channel (red, green and blue). Each bin describes the relative frequency in which that color intensity interval appears within a blob. When comparing two normalized histograms, the Bhattacharyya coefficient, B_c, computes a similarity value between them. This coefficient is simply the ratio of overlap between two histograms. It can be computed as follows, for two histograms H_1 and H_2 with n bins:

$$B_c(H_1, H_2) = \sum_{i=0}^{n} \sqrt{H_1^i H_2^i} \tag{8}$$

where H^i is the value of the i^{th} bin in the histogram. For a detected blob d_j and track T_k the color similarity function, S_c, is:

$$\mathcal{S}_c(d_j, T_k) = B_c(H(D_j), H(T_k)) \tag{9}$$

where $H(d_j)$ is the histogram computed at the pixels of the blob mask and $H(T_k)$ is the histogram of the last blob associated with the considered track.

Data Association - We then attempt to associate each existing track with a detected blob. Consider the set of active tracks at time-step $t-1$, $T^{(t-1)}$, as well as the set of detected blobs $D^{(t)}$. A track T_k will be associated with the blob D_j when:

$$\mathcal{S}_p(d_j, T_k) \geq \theta_p - \gamma_{p,T_k} \tag{10}$$

$$\mathcal{S}_c(d_j, T_k) \geq \theta_c - \gamma_{c,T_k} \tag{11}$$

$$k = \text{argmax}_\text{i}(\mathcal{S}_c(d_j, T_i)) \tag{12}$$

where θ_p and θ_c are minimum thresholds for each similarity function. To improve the behaviour of the associations, we add a decay, γ_{p,T_k} and γ_{c,T_k}, to the minimum similarity thresholds required. This decay is based on how long an existing track has not been matched with a blob. This helps when a tracked fish significantly changes in appearance, or moves in an unexpected way such that the predicted position is not accurate.

Local Search Recovery Method - After every detected blob has been considered for association with a given track there is still a chance that it has not been matched with a detected blob. This, however, does not guarantee that the tracked fish isn't present in the scene and can occur when the detection module fails to identify the presence of a fish. So when a track is not associated with any detected blob a recovery tracking method is applied. This method consists in building an appearance heat-map around the predicted position of the tracked fish. For each pixel in the area surrounding the predicted position the estimated probability of the color fitting the tracked fish color histogram is computed. This is done with the color histogram of past blobs for that track. Then each pixel is flagged as either positive (meaning it probably is part of the fish) or negative, using a threshold parameter. A two-pass connected component algorithm then removes isolated false-positives and identifies the biggest connected region of positive pixels. If the area of this region is larger than $\frac{3}{4}$ of the area of the last blob then it is associated to the track as the blob for that frame. A given track, however, cannot repeatedly use this recovery method alone. This would cause a drift of the color histogram for that track, possibly resulting in the repeated tracking of a background region.

Track Lifespan - Detected blobs that were not matched to an existing track, through the method described in section 4, get assigned to a new track. Finally, tracks that have not been associated with a blob (detected or recovered) for longer than 2 seconds (or equivalent number of frames when running the system in frame-by-frame mode) are deleted. This happens when a fish leaves the scene or is occluded behind an object.

5 Classification

For fish classification is required a collection of sample photos for each species of interest. For each species were collected about 20 examples, with the background removed.

Coral Reef Scenario One of the environments is a coral reef. In this environment, the colors are well defined, the water is clear and the depth is not a factor impairing visibility. These conditions allow us to classify the fish through color similarity metric, the Bhattacharyya coefficient.

Features - Consider two relative frequency color histograms, H_1 and H_2, computed from the colors of the pixels in two distinct blobs. If the overlap of these histograms is significant, the two blobs probably belong to two fishes of the same

species. This assumption has drawbacks which are discussed later. The function that describes this overlap is the Bhattacharyya coefficient, B_c.
Classification - In order to classify a track we consider the average Bhattacharrya coefficients, B_c, of the last $k \leq 5$ blobs of that track and each of the species images within our dataset. The species set with the highest average coefficient becomes our classification for that track. This simple classification method proves reliable when the colors of the species are well defined and there is not much overlap among the appearance of the present species. However, it does suffer in performance when lighting is not stable enough or the colors are washed out.

Main Tank Scenario The second environment of our test scenarios displays conditions that make it hard to distinguish fish species by relying on their color. First, due to depth, it wears out colors, leaving everything with a blue tint. Secondly, the dimensions of the tank allow fish to wander further from the camera and thus appear smaller and with less contrast. To overcome these two problems, a solution that does not rely on color cues or the scale of the fish is required. An histogram of oriented gradients (HOG) [2] fits these requirements, when treating images scaled to the same dimensions.
Features - An HOG describes directional changes in color, capturing features such as edges, that would otherwise not be considered. The magnitude and direction of the gradients are computed based on the vertical and horizontal per-pixel gradients. Then, the image is divided into blocks. The gradients within each block are normalized, which makes the features more robust to variations in brightness. The histograms of gradients are then computed on a per-block basis.
Classification - In order to compute the similarity between two images we use linear regression to reconstruct the HOG features of a detected fish based on the HOG features of each species dataset. The species dataset that best reconstructs the HOG features of the detected blob becomes the class associated with that blob. By down-scaling the sample images and blobs we are able to keep this method running in real time. This linear regression approach is based on the work in [7].

Let μ be the vector of HOG features of a given image. A set of n images belonging to a class dataset (indexed j), can then be concatenated into a matrix ζ_j, where the columns are: $\mu_1, ..., \mu_n$. Given a new image, i, whose class we want to identify we need to first compute its vector of HOG features, γ_i. Assuming that μ_i can be written following the linear hypothesis:

$$\gamma_i = \zeta_j \theta_j \tag{13}$$

where θ_j are the linear regression coefficients, we project the HOG features onto the class j feature subspace, as follows:

$$\theta_j = (\zeta_j^T \zeta_j)^{-1} \zeta_j^T \gamma_i \tag{14}$$

$$\hat{\gamma_j} = \zeta_j \theta_j \tag{15}$$

The difference between the values of γ_i and $\hat{\gamma}_i$ form the residuals, R_i^j for that image:

$$R_i^j = ||\gamma_i - \hat{\gamma}_i|| \tag{16}$$

A smaller residual value means the regressed feature vector is closer to the original than otherwise. To arrive at a final classification we cast an exponentially weighted vote based on the residuals R^j for each species image set j. Each residual, for image i, R_i^j, casts an exponentially weighted vote support class j:

$$\omega_i^j = e^{-\beta R_i^j} \tag{17}$$

Then, we select the class j, with the maximum summed vote weight:

$$c = argmax_j(\sum_i \omega_i^j) \tag{18}$$

Temporal Consistency - Regardless of which of the two methods used, to further improve the stability of our classification module we take the most frequent classification of blobs in a track for the previous $n = 20$ time steps. We allow our blob classifiers a significant error margin while keeping the overall track classification accurate.

6 Evaluation

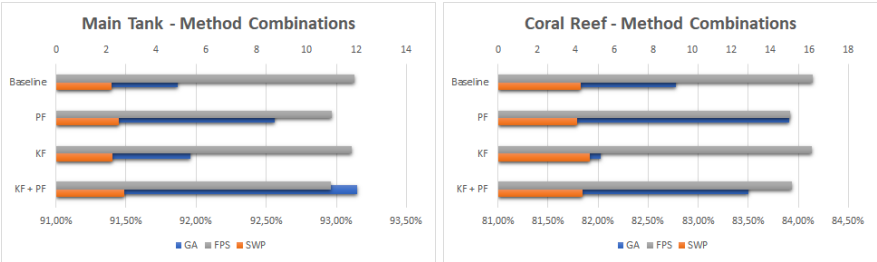


Fig. 4. Impacts of the proposed methods in each environment. GA - Global accuracy; FPS - Processed frames per second; SWP - Number of identity swaps.

Detection and Tracking - We tested our system with and without Kalman Filter based prediction & Local Search Recovery on both scenarios, Fig. 4.

On the main tank environment, as expected, our system performs better with both methods enabled. Looking at them individually we see that the Kalman Filter position prediction does not have a considerable impact on its own. Similarly, the Local Search Recovery Method only seems to improve the global accuracy by 0,6%. When used together, these methods show a slight (1,4%) improvement on the global accuracy.

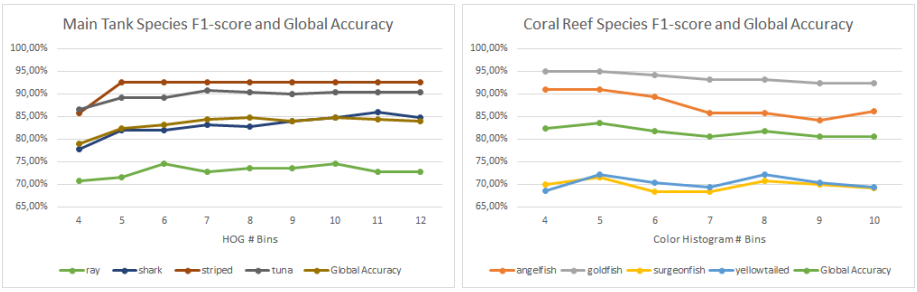


Fig. 5. Global Accuracy and F1-Score for the species in the Main tank (left) and Coral Reef (right) scenarios.

Fig. 6. Part of the ray classification dataset.

On the coral reef scenario we see that, while the recovery method improves the global accuracy, the Kalman filter position prediction method seems to affect performance negatively with and without the recovery method enabled. This can be attributed to the more random nature of movement presented by the smaller fish in this environment. The metrics of the tests in the coral reef are overall worse than on the main tank for two reasons: the fish remain on scene longer, meaning that target losses are expected to be higher, as there are more opportunities for missing detections and other issues such as crossing fish; the *yellowtailed* fish are so small that they sometimes remain undetected for long periods of time, explaining the few mostly lost tracks and the overall lower global accuracy.

Classification - In the case of the main tank, in Fig. 5, the F1-score of the ray species is significantly lower than the rest. Rays have a unique way of moving. They have "wings" which flap in order to swim through the water. This way of swimming results in considerable deformation of their bodies as they present a wide range of shapes, as seen in Fig. 6. This explains why a classifier based on shape features would present a lower precision value for the ray species.

On the coral reef scenario, in Fig. 5, we observe two values that stand out. The *yellowtailed* recall (56,5%) and the *surgeonfish* precision (56,9%). When considering blobs of these two species, the few background pixels around the smaller fish make up for a bigger portion of the blob than in the case of the *surgeonfish*, throwing off the expected color histogram. Although it might be

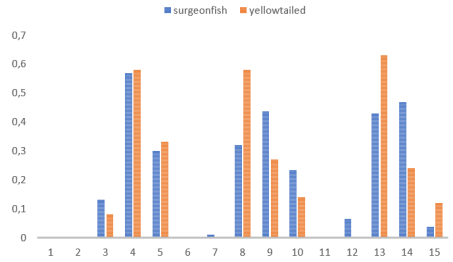


Fig. 7. Color histogram comparison between the blobs of a "yellowtailed" and a surgeonfish.

easy for us to distinguish between these two, specially due to their size, they are quite similar in color under certain lighting. Figure 7 illustrates how similar the histograms of the two species can be.

7 Conclusions and Future Work

We first discussed the motivation and benefits of tracking and classifying fish in real-time. Video tracking is a hot topic in the domain of computer vision, but that has not been discussed much in the realm of marine wildlife observation. Marine environments, and thus fish, are facing ever growing sustainability problems due to human interference. A proper application of video tracking and image classification could aid biologists in gaining a better understanding of marine wildlife and how they are impacted.

We proposed and implemented an approach to track and classify fish in real-time on a unique dataset built in the *Oceanário de Lisboa*. We implemented methods supporting the usual approaches that we deemed appropriate for our problem. These were able to deal with some of the shortcomings that are associated with tracking based on detection. We also implemented classification methods able to deal with the unique challenges presented by the different environments in our dataset.

The presented system is just one of the possible approaches to our problem. The biggest hurdle is dealing with the restraints associated with enabling our system to run in real-time. This heavily limited the feasibility of most methods. We also had to deal with the problem of having a limited dataset to work with, as most classification algorithms require large training data-sets for adequate results.

Some of the challenges presented remain unaddressed by our proposed solution. The main problem that affects our system is the overlap of fish. Solving it requires proper segmentation on a per-object basis which we were unable to achieve. Our approach to tracking, based in tracking by detection, is prone to unrelated moving objects. While we were somewhat successful in addressing the issue of repetitive background movement, which mostly appears due

to coral and algae, our system is still vulnerable to unrelated moving entities that could appear in other environments. While we successfully implemented classification methods for our defined problem, it was significantly simplified by grouping species with similar appearance. We would like to see a more complete discretization of fish species and a solution that is able to deal with that, significantly harder, problem.

8 Acknowledgments

This work was supported by Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020, Project E-ARK3 and partially supported by FCT with the LARSyS - FCT Plurianual funding 2020-2023.

References

1. Emma Beauxis-Aussalet, Simone Palazzo, Gayathri Nadarajan, Elvira Arslanova, Concetto Spampinato, and Lynda Hardman. A video processing and data retrieval framework for fish population monitoring. In *Proceedings of the 2nd ACM International Workshop on Multimedia Analysis for Ecological Data (MAED)*, pages 15–20, 2013.
2. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.
3. Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.
4. Nir Friedman and Stuart Russell. Image segmentation in video sequences: a probabilistic approach. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 175–181, 1997.
5. Isaak Kavasidis and Simone Palazzo. Quantitative performance analysis of object detection algorithms on underwater video footage. In *Proceedings of the 1st ACM International Workshop on Multimedia Analysis for Ecological Data (MAED)*, pages 57–60, 2012.
6. Isaak Kavasidis, Concetto Spampinato, and Daniela Giordano. Generation of ground truth for object detection while playing an online game: productive gaming or recreational working? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 694–699, 2013.
7. Uzair Nadeem, Syed Afaq Ali Shah, Mohammed Bennamoun, Roberto Togneri, and Ferdous Sohel. Real time surveillance for low resolution and limited-data scenarios: An image set classification approach. In *arXiv:1803.09470*, 2018.
8. Concetto Spampinato, Yun-Heh Chen-Burger, Gayathri Nadarajan, and Robert B Fisher. Detecting, tracking and counting fish in low quality unconstrained underwater videos. In *Proceedings of the 3rd International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 514–519, 2008.
9. Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31, 2004.