TÉCNICO LISBOA

Video Based Live Tracking of Fishes in Tanks

José Bernardo Nazaré Barbosa Castelo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Prof. Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Member of the Committee: Prof. João Paulo Salgado Arriscado Costeira

November 2019

Acknowledgments

I would like to write a few words of appreciation towards those that played a key role in making this thesis possible.

First, to my advisors, professor Sofia Pinto and professor Alexandre Bernardino, I am endlessly thankful for all the encouragement and guidance they have provided me during the past year. Their continuous effort not only made this thesis possible but also made me grow both as a student and as a person. I can only hope they have taken as much out of this experience as I have.

Secondly, an expression of gratitude towards Núria Baylina, who was our point of contact with *Oceanário de Lisboa*. I hope that further collaborations take place, enabling Núria and her colleagues to expand the available toolkit for their work at *Oceanário de Lisboa*.

Finally, I would like to thank those that have supported me daily during this thesis: my parents, Fernando and Ana, for allowing me this opportunity; my siblings, Rita and João, for all the advice; my colleagues, especially those also guided by professor Sofia, for all the discussions and feedback; my friends, for being there when it mattered the most.

Thank you.

Abstract

The application of video tracking has been a hot topic in surveillance and monitoring settings such as vehicle and pedestrian tracking in roadways. In this thesis we explore video tracking and classification in the context of real time marine wildlife observation. Among other applications it could help biologists by automating the process of gathering data, which in many cases is done manually. We review previous work done in this context, discussing the advantages and shortcomings of each approach considering the characteristics of our problem.

A system to tackle the challenge of tracking and classifying fish in real time is proposed and implemented. We apply Background Subtraction techniques in order to detect the fish, followed by Feature Matching methods to track their movements over time. Here we implement methods that are able to deal with some of the shortcoming of tracking by detection, based on a Kalman Filter to predict fish positions and Particle Filtering to recover undetected fish. The species of each tracked fish is also identified through Image Classification methods, using different features appropriate to the different environments.

We develop and test our system using a custom built dataset, recorded and labeled by us in collaboration with the *Oceanário de Lisboa*. The impact of the proposed improvements to the used tracking methods are quantified and discussed. The proposed system is able to track and classify fish in real time with satisfactory results in two distinct scenarios that provide different challenges.

Keywords

Object Detection, Video Tracking, Image Classification, Real Time, Fish

Resumo

A aplicação de seguimento de objetos em vídeos tem vindo a ser uma área de alto interesse no domínio da video vigilância como, por exemplo, no seguimento de veículos e peões na estrada. Nesta tese exploramos os tópicos de seguimento por vídeo e classificação no contexto da observação de vida animal marítima. Entre outras possíveis aplicações isto poderá a vir ajudar os biólogos ao automatizar o processo de recolha de dados, o que em muitos casos é feito manualmente. É feita uma revisão de trabalhos prévios neste contexto, discutindo as vantagens e desvantagens de cada abordagem tendo em conta as características do nosso problema.

É proposto um sistema capaz de seguir e classificar peixes em tempo real. São usadas técnicas de subtração de fundo para detetar os peixes, seguidas de técnicas de associação de dados para acompanhar os movimentos destes ao longo do tempo. Para este efeito, propomos métodos capazes de lidar com algumas das desvantagens do seguimento por deteção, baseados num filtro de Kalman para prever as posições dos peixes e em filtros de partículas para a recuperação de peixes não detetados. As espécies dos peixes são identificadas através de algoritmos de classificação de imagens baseados em características apropriadas a cada ambiente.

O desenvolvimento e testes do nosso sistema foram feitos com recurso a um conjunto de dados construido por nós em colaboração com o Oceanário de Lisboa. É medido e discutido o impacto dos métodos implementados para colmatar alguns dos defeitos do seguimento por deteção. O sistema proposto é capaz de seguir e classificar os peixes em tempo real com resultados satisfatórios em dois ambientes que apresentam desafios distintos.

Palavras Chave

Deteção de Objetos, Seguimento por Vídeo, Classificação de Imagens, Tempo Real, Peixes

Contents

List of Figures

List of Tables

1

Introduction

Contents

1.1 Motivation

Almost everything we know about animals comes from observing and analyzing their behaviours. There are many ways in which we can observe an animal in its habitat. In the past, biologists dedicated their life travelling the world in search of their subjects, spending a great deal of time in the wild observing and logging their findings. A limitation of this method is that the behaviours and even the appearance of certain animals is conditioned by the presence of a human, a foreign entity to most animal habitats. Additionally, the information conveyed by a biologist is subject to his/her interpretation of a phenomenon, not allowing for an objective portrait of a behaviour or trait displayed by an animal that can later be analyzed in a different light or spirit. Advances in technology allowed for more ways in which biologists are able to observe, and then study, their subjects.

The first big technological breakthrough was the appearance of the still camera. The still camera allowed biologists to take photos of their subjects, revolutionizing the process of cataloging species, which was previously done by manually drawing each animal. However, only so much information can be conveyed by a still photo. Acquiring a better understanding of complex behaviours and traits requires observing an animal for their complete, continuous duration/existence. The appearance of the film camera allowed taping video sequences of the animals in their habitat. This proved to be a powerful tool when collecting data, as it conveys more information than still pictures on their own by introducing the temporal aspect of video. Videos in their analog form (film) allow biologists to study a certain behaviour or trait in more detail, as they can replay scenarios which would otherwise only be observable in the field. 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 and drawbacks. Some of the advantages include:

- a non intrusive way of observing animals, unaffected by the presence of a human;
- 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 monetary cost, as cameras and computing power become more economically viable;
- allow for remote observation or analysis.

and some of the most relevant drawbacks consist of:

- the output of significant amounts of data, requiring considerable amounts of data storage capacity;
- the necessity of processing said output, as only small portions of the observations convey useful information;
- a limited view-port, in the sense that only what is in front of a camera is captured, potentially leaving out relevant information;

In order to gain a better sense for the work involved in wildlife observation and research we contacted our local marine wildlife center, the *Ocenário de Lisboa*. Here, efforts are made towards education, marine wildlife conservation and research. The *Ocenário 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 figure 1.1.

(a)

(b)

Figure 1.1: Two of the tanks at the *Ocenário de Lisboa*.

Part of the work done at the *Ocenário de Lisboa*, whether for maintenance, monitoring or research

purposes, involves close observation of their fish. Tasks such as measurements of fish (size and weight estimations) and species population surveys require that biologists manually gather this data. We identified that some of these tasks could benefit from video observations, as well as partial automation of the processing aspect that they entail. Thus, in collaboration with the *Ocenário de Lisboa* we filmed their tanks with the goal of creating a dataset that would allows us to develop and test a system which is able to automatically track and identify their fish. It is also in their interest to be able to physically observe certain situations where, for example, a given species is displaying a behaviour that is rarely seen. This motivated us to enable our system, which constantly monitors the observed tanks, to relay information regarding their fish in real-time. As a marine wildlife center that hosts visits to their exhibits, however, there are a few strict restrictions that must be respected. We must not, in any way, impact the visitors experience or interfere with the environment the tanks provide to the animals. Moreover, different environments within the tanks of the *Ocenário de Lisboa* have distinct properties that provide us with different challenges:

- the depth of each tank influences the perceived colors, as seen by the blue tint in figure 1.1(a);
- the size of the fish greatly impacts their behaviour, with smaller fish displaying faster and more unpredictable movements;
- the uniformity of certain groups of fish species makes distinguishing them in videos an extremely difficult task;
- the size of certain tanks, as seen in figure 1.1(a), limits the visibility of fishes that are swimming at considerable distances from the camera;
- the presence of marine flora poses a challenge in detecting fish, due to their movement and provided camouflage, as seen in 1.1(b);
- variations in lighting present a constantly changing environment;
- tank features, such as the rocks seen in 1.1(a), conceal the fish swimming behind them;

1.2 Problem Statement

Detecting and tracking the position of other entities is something that every animal does out of necessity. In the predator-prey relationship detecting the opposing animal is a life or death matter. Most animals relying on vision to detect the position of others animals display a common behaviour: they fixate their sight looking for movement in their environment, which is a clue for the presence of another animal. Once the presence of another entity is detected the animal must recognize whether it is a threat or a

prey. This can also be thought of as a classification problem based on the animals sight and previous knowledge.

Birds from a species family known as Kingfisher spend a great deal of time looking for fish or small insects in the water. They stabilize their heads while watching a body of water, patiently waiting for the movement of fish or insects in order to catch them. The same concept can be applied in order to track fish in videos. By keeping a camera in a still position we can detect the presence of fish, using background subtraction techniques, allowing us to follow their movements and identify their species using image classification.

Video tracking is the process in which we follow the path taken by target objects in image sequences. Regardless of the tracking method, in order to follow the location of objects, we must first be able to detect their presence. This is known as object detection. When tracking multiple target objects we must also be able to distinguish the objects from one another. This usually involves extracting and comparing descriptions of their traits, more commonly known as features.

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 F_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 which species, c_i out of the species set $C = c_0 : c_N$, each tracked fish T_k belongs to.

A few restrictions and assumptions have to be taken into consideration, based on our environment:

- the scene is captured by a statically positioned camera, from outside of the tanks. We are restricted to this option since we may not interfere with the fish in the tanks or the experience of the *Oceanário de Lisboa* visitors. This is also significant to the feasibility of some detection methods that rely on the fact that the camera is statically positioned relative to the captured scene.

- the resolution of the video output by the camera must be at least *720x480p*. A lower resolution means there is not enough detail to the analyse scene. This heavily limits the number of pixels on targets, and thus the distance at which we are able to track and classify the fish.

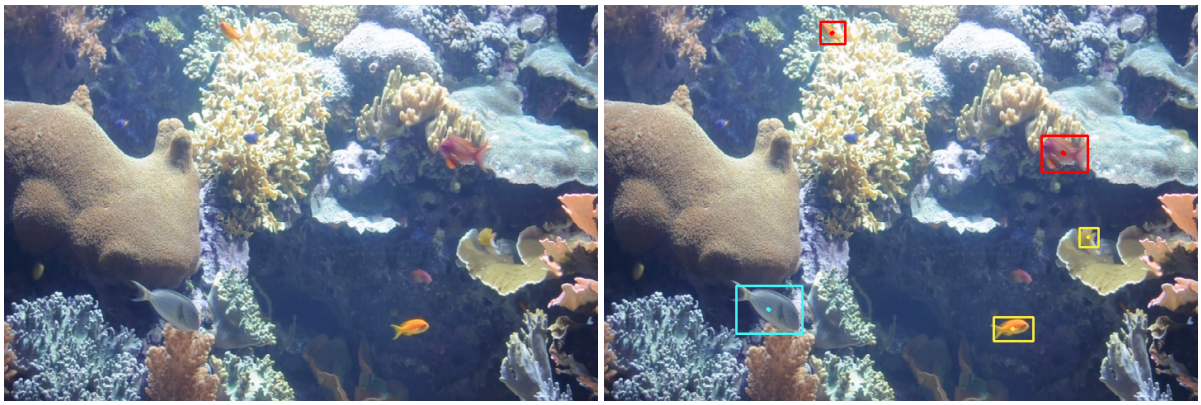
- when classifying the tracked fish, our system requires a dataset for each species of interest. While this means that each new environment requires manual setup we also provide a way to build this dataset by running our system with the classification disabled.

- a closed environment is assumed, meaning that the possible species of the observed fish will not change from those originally in the system. This is true for controlled environments such as the tanks in the *Oceanário de Lisboa*, and, while it does not allow us to generalize our results to open environments, it significantly simplifies the classification problem.

1.3 Objective

We aim at improving fish observations through the use of videos, whether that be 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.

The formal objective of our thesis 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 figure 1.2(a), where the desired output is the location of each fish (highlighted with a bounding box) and species (bounding box color), and are exemplified in figure 1.2(b).



(a) Example frame **(b)** Desired output

Figure 1.2: Goal of the thesis: live feed of locations and species for each fish in a tank.

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 keep the system running in real-time on ordinary computers. This makes our system economically viable for most people, with no extra investment necessary into specialized hardware.

1.4 Contributions

We propose and implement one possible approach to tracking and classifying fish in real-time. Our approach is based in well-known methods in their respective areas but that, as far as we know, have not been previously used together in the context of aquatic wildlife observation. The real-time support allows for continuous observation of an environment, which is highly desirable for monitoring and surveillance purposes. Additionally, our system is subject to being adapted and used in other similar scenarios, such as land animal observations, for example.

The detection of fish is achieved using background modelling and subtraction techniques, namely an Adaptive Gaussian Mixture Model. This technique is able to deal with background movement of unrelated entities in the scene which could interfere with the detection of the objects of interest.

Similarity measures and data association methods are used in order to track the detected fish. We also implement a method of dealing with some of the flaws of tracking-by-detection, based on particle filtering methods.

Image classification methods are used to identify the fish species. We provide classifiers based on linear regression techniques for fish shape features as well as color similarity features. These present solutions that are suited to the different environments, with distinct characteristics and associated challenges presented by the environments at the *Ocenário de Lisboa*.

Our system successfully tracks and classifies most of the fish in two distinct scenarios taking place at the *Ocenário de Lisboa*. We overcome a range of challenges provided by these environments while providing a non intrusive solution, both to the fish and to the visitors.

1.5 Outline

In chapter 2 we first introduce base concepts of the domains related to this thesis in section 2.1. A review of the state of the art methods, as well as past work with similar settings and goals are presented in section 2.2. Chapter 3 first provides an overview of our system architecture, seen in section 3.1, with a detailed description of our solution to each sub-problem as well as how different challenges are addressed in sections 3.3, 3.4 and 3.5. Our test setup and the dataset built in collaboration with the *Ocenário de Lisboa* are described in section 4.1. The test methods and considered evaluation metrics are formulated in section 4.2. The results of these tests are displayed in section 4.3, with an analysis and discussion presented in section 4.4. Finally, we discuss unaddressed problems and future work in section 5.2.

2

Background and State of the Art

Contents

In this chapter we present some of the work related to the theme of this thesis, in the areas of object detection, tracking, and classification. Figure 2.1 presents a typical approach to our problem. In the following section we review some of methods used in each of the steps of this pipeline, some of their advantages and drawbacks, and how they might help us solving our problem.

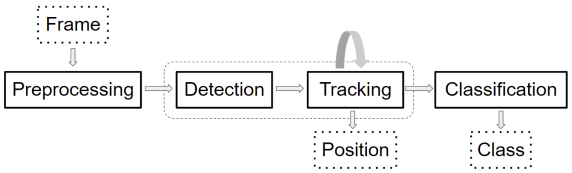


Figure 2.1: Typical approach to the tracking and classification problem in videos.

2.1 Background

Before reviewing the state of the art we introduce key terms to readers unfamiliar with the areas of computer vision and artificial intelligence. In this section we present some aspects of the three main domains discussed in this thesis: object detection, object tracking and image classification.

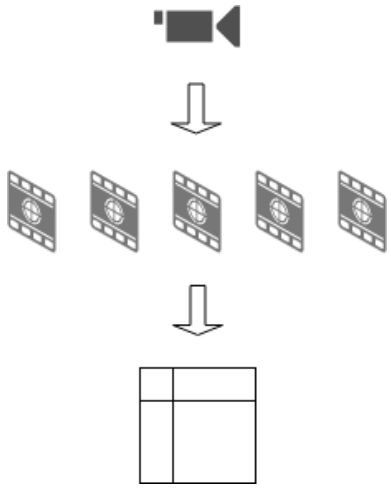


Figure 2.2: Structure of a video file.

2.1.1 Video Data Structure

First, it is relevant to think about the way our subject data, videos, are stored and processed. The overall structure of a video can be seen in figure 2.2. A video is composed of a sequence of images, each known as a frame, F. The amount of frames recorded per second is known as the frame-rate, r, of a video. This means that a video with a duration $T = 10$ seconds, with a frame-rate of $r = 25$, will be

composed of $r \times T = 25 \times 10 = 250$ frames. Each frame $F_1, ..., F_{250}$ in this video is composed of a matrix of pixels. The dimensions of these matrices are known as the video resolution. A video resolution is composed of a width and a height, with the units being pixels (the elementary unit of a frame, short for picture element). If our video has a resolution of 720×480 pixels (frame width and height respectively), then each frame F_t is a matrix with 720 columns and 480 rows, where each cell contains the RGB value of the respective pixel. In figure 2.4 a portion of a frame is zoomed in, where the matrix like structure can be seen. An RGB value is a description of a color, it is a vector with 3 components: the red value, the green value and the blue value. Each of these is within the integer interval $[0...255]$. This means the RGB color range is able to describe $256^3 = 16.777.216$ different colors. The RGB color space can be seen in figure.

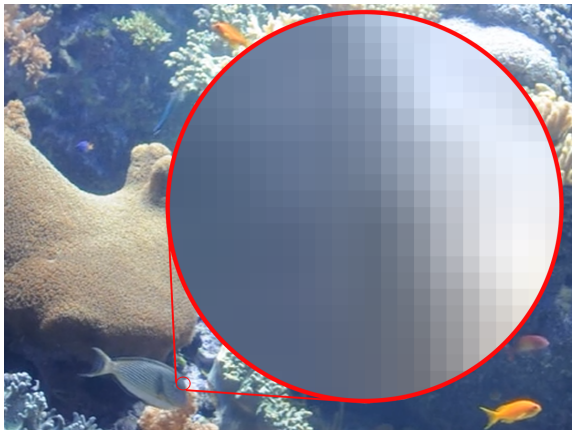


Figure 2.3: Pixels of a frame, exemplified.

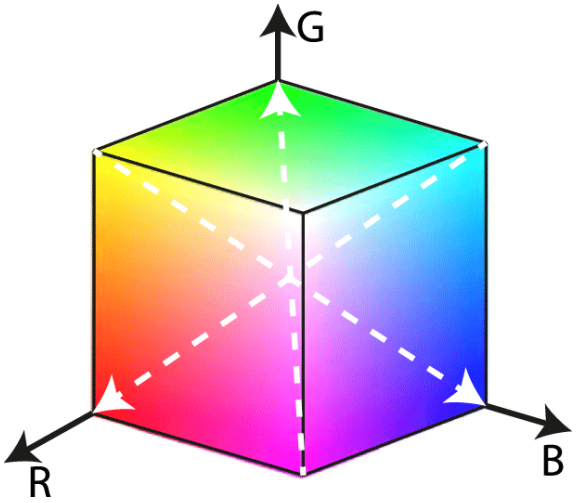


Figure 2.4: The RGB color space.

2.1.2 Detection

Object detection is a form of video content analysis, in the realm of computer vision, that aims at locating objects of interest in video sequences. Video motion detection is the more commonly seen form of object detection. This form of detection relies of stable video footage with a static background in order to identify the objects based on their physical movement. Traditional forms of moving object detection include:

- **Frame Differencing**, where the difference between two consecutive video frames are considered.
- **Temporal Differencing**, where a per-pixel difference function of a set of consecutive video frames is considered.
- **Optical Flow**, where the apparent velocity of color patches in consecutive video frames allows detecting moving objects.
- **Background Subtraction**, where a statistical model of the background allows extracting the foreground regions of video frames.

The other form of object detection, which is template based, relies on computing visual object features of video frame regions in search of feature values similar to those of the objects of interest. For this type of approach there are two families of methods:

- **Machine Learning**, where the visual features must be explicitly formulated and then applied to learning algorithms, such as a Support Vector Machine.
- **Deep learning**, where the visual features are not explicitly formulated but learned and extracted by neural networks, such as Convolutional Neural Networks.

The biggest difference between motion based detection and template based detection is that the former does not consider the appearance of an object but relies on its movement. For this reason one is not objectively better than the other, but each is better suited for distinct scenarios. One factor that dictates the feasibility of detecting objects based on movement is whether the camera recording the scene is static (known as **onshore**) or presents some movement relative to the background (known as **offshore**). If a camera is offshore then the background will contain movement between frames and detection based on object movement becomes less feasible.

2.1.3 Tracking

Video tracking aims at following the location of objects of interest during frame sequences. This usually entails:

- **object representation**, having a description (known as model) of the objects of interest.

- **object localization**, being able to detect the position of the object of interest.
- **data association**, being able to associate objects of interest between consecutive frames.

The order in which these are applied is not strict, while some tracking algorithms rely on the object representation in order to perform their localization (usually referred to as **template based tracking**), other algorithms do the opposite (**tracking by detection**). Some tracking algorithms take into account the dynamics of the movement presented by their targets, improving their object localization and data association methods. This is specially useful when we known model of the movement presented by the target objects.

The data association portion of tracking aims at relating detected objects in consecutive frames according to their identity, based on their properties. This usually involves comparing features such as location, size and other appearance based metrics. The **Hungarian Method** (formally named Kuhn-Munkres algorithm), and the **Nearest Neighbour Filter**, for example, are two commonly seen methods used for the data association portion of video tracking.

2.1.4 Classification

Classification is the problem of identifying which class an instance object belongs to, out of a set of possible classes. In this thesis we refer to classification in the context of Machine Learning, a sub-area of Artificial Intelligence.

Dataset

A classification problem begins with a dataset. A dataset is a set of samples (also known as **instances**). A dataset whose samples are associated with classes are known as labelled datasets. A labelled class known to be accurate is also known as being the **ground-truth** class.

Datasets are usually divided into **training** and **testing datasets**. The purpose of this division is to avoid testing a given algorithm with the same dataset used in order to train it. Otherwise, an algorithm is subject to **overfitting** its training dataset, over-learning certain properties that might only be true for that given dataset, resulting in an inability to performance as well when given new samples.

Machine learning paradigms

In the realm of Machine Learning there are three major types of learning paradigms:

- **Unsupervised learning** algorithms group unlabeled values together based on their properties. These groups are known as clusters and represent a class. A new observation is then associated with a given cluster, based on its properties. This process is usually done given a distance function

between two values and is known as **Clustering**. Clustering algorithms minimize the distance among the members of each cluster, grouping similar values together.

- **Supervised learning** algorithms require a set of labeled values. Based on the properties (also known as features) of each class, and similarity metrics, supervised classification algorithms provide a mapping from observations to classes. Within supervised learning algorithms there is also the problem of regression, where, instead of a single class output, a probability of the observation belonging to each given class is computed.
- **Reinforcement learning** algorithms learn by trial and error. A labeled set of values is not directly provided to the algorithms. Instead, they learn by example: taking a decision, observing the outcome and adjusting their behaviour in the following iteration.

Parametric and Non-parametric Algorithms

Regardless of the paradigm, there are **parametric** and **non-parametric** algorithms. Parametric algorithms assume that the observations within dataset fit a given model and only requires finding the values for the parameters the model calls for. On the other hand, non-parametric algorithms do not assume a given model but rather learn the model that a dataset presents. This means that while parametric algorithms usually learn and run faster, they present worse performance when the model does not suit the dataset when compared to their non-parametric counterpart.

2.2 State of the Art

The *Fish4Knowledge* Project

By far, the most extensive project in regards to marine wildlife video observations is the *Fish4Knowledge*. This project was funded by the European Union Seventh Framework Programme and took place over 3 years with the collaboration of 27 project members. *Fish4Knowledge* aimed at creating a system for collecting, storing and analyzing remove fish observation video data. The main motivation behind this project was to provide a system to gain a better understanding of marine wildlife environments and their dynamics. The formal objectives of the *Fish4Knowledge* project were:

- Detect targets in noisy underwater environments.
- Recognize target species.
- Characterize interactions between targets.
- Store and allow timely access to the collected video data.

- Provide a web interface to query the extracted data.

The main challenges of the *Fish4Knowledge* project, technical and otherwise, were:

- Overcoming the issues presented by the marine environment, such as the presence of floating debris in the water, the build up of algae in the camera lenses, the varying lighting depending on the weather and time of the day.
- Handling the sheer amount of data generated by the cameras (amounting to over 200TB by the end of the project).
- Learning and integrating domain knowledge specific to Marine Biology into their solution.
- Identifying target species based on video data alone, which sometimes requires dissecting fish and analyzing their bones.
- Conveying project results in a concise and public platform, according to existing open source fish ontologies.

As a testbed for developing a system that is able to achieve the specified objectives and deal with the associated challenges they employed remote underwater static cameras at 10 different marine sites around Taiwan, continuously recording and uploading their video feeds to a central database. The framework of how data is retrieved, stored, processed and distributed is discussed in-depth in [3] and [22]. Labelling such huge amounts of data required building multi-user collaborative ground-truth labeling tools, such as an interactive game in which the users label ground-truth target segmentations and species for short video segments, discussed in [17], [18] and [9]. A semi-automated labeling method based on a nearest neighbour search of visual properties(contour shape and colors) was also proposed in [5]. Labelling the collected data allowed tests of different detection methods in [16], as well as their own systems performance. A web interface was developed, allowing remote access to live and past video feeds, as well a way to query the processed historical data. Some of the publications associated with the *Fish4Knowledge* project, which are directly related to the objectives of our thesis, are discussed next.

In [25], the authors discussed the automatic analysis of fish video observations in unconstrained environments collected as part of the *Fish4Knowledge* project. In order to detect fish, based on its movement relative to a static background, a moving average (MA) background subtraction algorithm was applied. Given a background reference in time-step t, B_t, and a frame, F_t, the MA flags pixels as belonging to the foreground (or in other words, belonging to a moving object of interest) if the absolute difference in value of a given pixel between images F_t and B_t is greater than a given threshold T_a.

$$|B_t(x, y) - F_t(x, y)| > T_a \tag{2.1}$$

Realistically, as the background is changing, B_t cannot be constant over time. As the background changes, B_t is updated every frame, with a weighted sum between B_{t-1} and F_t, controlled by the learning parameter α that specifies how fast the background adapts to the current image.

$$B_t = B_{t-1} * (1 - \alpha) + F_t * \alpha \tag{2.2}$$

However, this is not enough to stop detecting repetitive movement from moving objects. In order to overcome this problem the authors filtered the false positives through the use of Adaptive Gaussian Mixture Models. The choice was a balance between not having to carefully calibrate any parameters and being computationally lightweight enough. An Adaptive Gaussian Mixture Model [12] (AGMM) is able to describe multimodal variables such as the intensity value of a pixel that is seen when a repetitive background motion is present (e.g.: an aquatic plant waving in the water).

After modeling each pixel of the background with an AGMM the authors classify each pixel of the current image based on the likelihood that the respective value fits the model through the Mahalanobis Distance [8] between the observed value and each of the Gaussian components. Using the output of the AGMM alone would result in slow-moving objects of interest being undetected. However, by intersecting the output of both algorithms, the authors arrived at a solution with good results since the shortcomings of each approach are overcome by the other.

$$SelectedPix(frame) = MovingAverage(frame) \cap FitAGMM(frame) \tag{2.3}$$

This output is also subject to a size filter, ignoring portions that are deemed too small to be part of an object of interest. Then, through the analysis of connected components (discussed in [10]), the number of fish present in the image is computed, as well as the bounding box for each fish, referenced hence forth as a blob.

Tracking was carried out by feature matching each blob to the previously known tracks. Feature matching consists in checking if the difference between features of a blob and the values stored in a track is less than a given threshold. Each feature has its similarity function and its threshold. In this case features used included: area, speed and orientation of each blob. Objects that are not matched with a track are assigned a new track. Analogously a track that did not match an object will be forgotten after a few frames of being continuously unassigned. Tests of this approach show that it is able to track about 85% of the fish in a given video. Furthermore, the authors mention that most failures would be handled with better fish identification approaches, as most errors occur during situations such as fish overlapping.

Identifying which species a given fish belongs to based on visual cues alone is a task which was approached in [14] by taking in consideration existing domain specific knowledge. A way to solve multi-

class classification problems is to use decision trees which, one level at a time, identify features that separate certain classes from the others. The authors implemented a hierarchical classifier, which is a decision tree that considers different features at different levels. In order to decide which features would better suit the different levels domain specific knowledge was incorporated by taking into consideration the taxonomy of fish. Taxonomy is the way in which we classify things. In the case of fish we classify them based on certain traits, which in this case are included in the considered features. By incorporating this knowledge when making a hierarchical tree the authors were able to add 4% to the accuracy of a baseline tree method, successfully classifying 90% of the fish in a test dataset.

In [24] a rule based event detection system for the *Fish4Knowledge* is proposed. As the trajectories and properties of detected fish are stored in the database they are also processed by a set of user defined rules (through their systems web interface) that trigger events under specified conditions. This allows marine biologists to automatically gather video samples of behaviours that they deem of interest.

To gain a better understand of which metrics marine biologists are interested in, when observing fish, and how the system implemented in the *Fish4Knowledge* project is perceived, different research teams were invited to test the system in [2]. Based on a semi-structured interview and interactions with the system the expert teams were asked regarding their understanding of how the system works and whether they trusted it. It was found that while most teams did not fully trust the output of the system, they would consider its metrics during their research if alternative statistics were also presented as an alternative.

Detecting faces in images

One of the most impact-full contributions in the area of visual object detection was the work by Paula Viola and Michael Jones in [29], published in 2001. In this publication they presented a machine learning approach for real-time visual object detection, introducing three concepts that are used to this day:

- The first contribution was the introduction of a concept they named Integral Image. An integral Image is the sum of grey-scale pixel values from the top left of an image to a given pixel location. This is a powerful concept in two ways: it allows for descriptions of region based features present in images, based on their brightness, and is relatively inexpensive to compute.

- The second contribution was a learning method based on *AdaBoost* (first introduced in [11]) which is able to select only the most relevant features (regions in their case) for distinguishing positive examples from negative examples in a given dataset. This allows a classifier to target the features that are most important to its decision instead of having to compute and consider all possible features.

- The last contribution by Viola and Jones in their publication was the introduction of the "*focus-of-*

attention" concept in classification methods. This concept aims at spending more computing power where appropriate. This means using layers of increasingly complex classifiers (considering more features, in their case) for more difficult examples where a simpler classifier wont be able to decide with a high degree of confidence. By applying this method their approach is able to quickly discard background regions of images and spend more time distinguishing the more challenging regions in images, allowing them to detect faces in real-time scenarios with competitive accuracy.

These three contributions were showcased by Viola and Jones in a face detection algorithm, vastly outperforming the existing methods at the time in both speed and accuracy. A big part of their success is based on how inexpensive and information rich the Integral Image concept is. For a given image i where $i(x, y)$ is the grey-scale value of the x^{th} left-most and y^{th} top-most pixel, the integral image value for that pixel, $ii(x, y)$, is computed as:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{2.4}$$

the integral image value allows for efficient computation of rectangle features for image sub-regions, which are still seen used today to a great extent. While the amount of information represented on each single region is not conclusive, the relationship among regions specific to certain structured patterns, such as faces, allows us to reduce the amount of regions needed to detect them. This fact is seen exploited to great success by Viola and Jones when employing progressively more complex classifiers that consider a larger number of regions depending on the complexity of a given image. One of the biggest downsides to the approach proposed by Viola and Jones is that such features are rotation variant, meaning that only targets aligned with the ones in their dataset samples will be successfully detected.

Detection and Classification, a comparison between traditional methods and Neural Networks

The authors of [28] compared the performance of two supervised machine learning approaches to detect and recognize coral fishes in underwater videos, with the main goal of surveying different species population pools in a given region. Much like other underwater video scenarios, problems such as color and lighting variations as well as sediments or floating debris (implying a moving background) had to be taken into consideration.

One of the tested approaches was the use of a Histogram of Oriented Gradients [7] (HOG) with a Support Vector Machine [27] (SVM). A HOG describes an object by computing the histogram of directional changes in intensity (oriented gradients) for each region of a given image. As such, it contains information derived from the edges of objects causing sharp changes in pixel intensity. This is a useful abstraction of the shape of an object, since it mostly disregards the actual colors of an object but focuses

instead on the change in colors. A SVM is a supervised method able to classify feature vectors (such as a HOG) by separating each class of interest, as well as possible, in a high dimensional space.

The second tested approach borrows from the field of Deep Learning. A neural network was used to classify images for each species. It mimics the way a brain functions by having neurons fire signals to other neurons based on their received signals and an activation function. The weighted signals received by each of them are evaluated by an activation function that decides whether to further spread the signal or not. In order for this network to be useful it has to be trained, adjusting the weights as needed until the outcome is correct for the appropriate input (such as outputting the class goldfish when the image contains a goldfish in it). To compute the appropriate weights for the neural network the authors used a technique called back-propagation. It slowly adjusts the weights of a network by comparing the expected output with the actual output when running known examples through the network.

Instead of using a simple network such as described above the authors decided to use a more complex one called Convolutional Neural Network [19] (CNN) which allows the network to extract the best features on its own. It works by using convolutional layers, which, by using a convolutional kernel morphs the signal. Then it pools the signals (usually by region) which reduces the dimensionality problem and improves performance by filtering the most relevant bits of a region (such as the maximum value, for example). The last convolutional layer will eventually output a value, representing the probability of the image belonging to each class, based on the summary of the most relevant information already gathered by the previous chain of layers.

In both approaches, regions of the image (frame of a video) are first given a motion score, which is computed based on the average absolute difference with the previous respective region. Each region is also ran through the recognition algorithm and assigned a probability of belonging to a class. Regions with a probability of belonging to a species above 98% are kept. All that remains is to check if multiple overlapping regions are identifying the same fish. Which is achieved by suppressing the bounding box with the lower probability when boxes have an overlapping ratio above 30% and share the same species. Results in this paper show that the Deep Learning approach has a better overall performance.

Tracking and classifying pedestrians

The work developed in [15] regards tracking and classifying vehicles and pedestrians through videos of an open environment with a static camera. The pipeline proposed by the authors is composed of three distinct modules: motion segmentation, where moving objects are detected; object tracking, where each individual object is tracked as time goes; object classification, where each object is classified as being a person, a vehicle, or other.

In the motion segmentation module foreground pixels are first identified by comparing the difference in values of the frame in question with the background reference image, according to a threshold (de-

termined experimentally), similarly to the work in [25], previously reviewed. The background reference image is built and updated with an Approximated Median Filter [21] (AMF). This filter computes the approximated median value within a neighborhood of a pixel and nudges the pixel value of the background reference towards that value. Authors mention that adding a step factor to that change according to the difference magnitude between these two values, such that the value adjustment is higher for bigger discrepancies, improves the results. Once the foreground pixels are identified they are segmented into separate objects by using a two-pass connected component labeling method, which identifies clusters of foreground pixels belonging to the same object of interest. Finally bounding boxes for each object are built based on the clusters obtained from the connected components.

The object tracking module of this project is split into four steps. First an object color model is built by computing a normalized RGB color histogram (count of each color value within the bounding box of each object). Then the position of the object for each previously existing track is predicted by a Kalman Filter [13], which, based on the position history at each time-step attempts to account for errors and provides a new estimate position. The velocity of each object is assumed to be nearly constant but noise is accounted for. The third step is an A-Priori Assignment of each object to a track. For each existing track (from previous frames) and detected object (from the segmentation module) a distance matrix is built by computing the distance from each predicted position (based on the track, from the previous step) and the actual position of each detected object in the frame. By checking which cells contain the minimum value in each row and column we are able to pair objects with existing tracks, storing their correspondence value (the Bhattacharya distance [1], which is a correlation value between the RGB histograms of the tracked object and the detected object). Objects are assigned to existing tracks if their correspondence value is above a certain threshold.

A last step merges tracks that are too close (overlapping objects), splits tracks that were merged and whose objects have now diverged in position, assigns new tracks to unmatched objects (when a new objects enters the scene) and tracks that remained unmatched for too long are forgotten (when an object leaves the scene). In the object classification module the authors have opted to classify each object based on its repetitive motion (defined as repetitive changes in shape) since walking pedestrians show this kind of motion whereas a vehicle is mostly static in shape. For each new frame at time-step t, and after accounting for changes in bounding box sizes (scaling operations are applied) a new Difference Image (D_t) is computed for each tracked object, i, as an exclusive-or operation between the values of pixels of the previous bounding box (O_{t-1}) and the current one (O_t).

$$D_t^i = O_{t-1}^i \oplus O_t^i \tag{2.5}$$

This means that a D will contain ones in pixels which have changed in value (accounting for noise with a filter threshold) and zeros in pixels which have not. A Recurrent Motion Image is the sum of all D for a

given object, in a given time interval with n frames. It has higher values in regions that change the most.

$$RMI_t^i = \sum_{k=0}^{n} D_{t-k}^i \tag{2.6}$$

Finally, a Motion History Image [4] (MHI) is computed by giving regions (block of pixels) that have an average value higher than a threshold (computed experimentally) a value of one and the rest a value of zero. A Motion History Image represents the areas that an object has occupied with its shape within its bounding box. In order to classify each object the Repeated Motion Ratio (RMR) metric was used. This is the ratio of blocks with a value of one in a given objects' MHI. A higher RMR means that there is a higher degree of repetitive motion (which, as previously stated, vehicles seldom have, and pedestrians have a lot). Thus, each object is classified as a person if its RMR is above an experimentally computed threshold or as a vehicle otherwise.

This pipeline was able to run in real-time with a reasonable CPU for today's standards. It achieved 33-50 frames per second on video feeds with a resolution of 756 x 576 RGB pixels. Furthermore, on a variety of sequences, each consisting of 600 to 1000 frames, it was able to correctly classify 38 out of 39 pedestrians and 20 out of 20 vehicles.

An adaption of Gaussian Mixture Models for real-time scenarios

The work in [26] proposes a real-time adaptation process for the Adaptive Gaussian Mixture Models reviewed before. With the use of approximations, the authors are able to more suitably model the background of complex scenes while keeping the computational complexity low enough to allow use of this technique online. Instead of trying to compute the Gaussians that strictly model values pertaining to background colors for a given region the authors suggest computing Gaussians that model the most common values and then decide which are most likely to correspond to background colors, based on the persistence and variance of each Gaussian in a given mixture.

The need to use multiple Adaptive Gaussians (a mixture) arises when more than one surface (background or not) can be seen in a particular region over time. Given a set S of k Gaussians that each models ω_i, an estimated portion of the color values seen in a given pixel during n consecutive frames of a scene, the authors compute the probability of observing a value x, observed in a new frame at time-step t as:

$$P(x) = \sum_{i=0}^{k} \omega_i * \eta(x, \mu_i, \Sigma_i) \tag{2.7}$$

with η being the Gaussian probability density function, and Σ being the co-variance. Some accuracy is sacrificed by assuming that each color channel (red, blue and green) is independent in value, in order to avoid computationally expensive matrix inversions.

In order to keep the weights ω_i updated a learning parameter α controls how fast they are adjusted. This adjustment is made according to the following formula:

$$\omega_{i,t} = (1 - \alpha) * \omega_{i,t-1} + \alpha * M_{k,t} \tag{2.8}$$

where $M_{k,t}$ is one if the current pixel value is matched with that distribution or zero otherwise. A match occurs when the value is within $T_m = 2.5$ standard deviations from the mean of a distribution. This means that the model adjusts exponentially to changes in distributions and that sporadic erroneous changes are also corrected faster. To decide which distributions describe the background they are first ordered by their value of ω/α, which is higher as the distribution describes a higher amount of values and has less variance. Then the first J distributions are chosen as background distributions (the authors name this technique Background Model Estimation) such that they describe T_r , the minimum portion of data to be taken into account as background:

$$J = argmin_j((\sum_{k=1}^{j} \omega_k) > T_r) \tag{2.9}$$

Repetitive motion of background entities could result in a larger value of J since it takes a bigger number of distributions to model all colors correctly. A pixel is considered as background only if it matches one of the selected Gaussians. Similar to previous reviewed work, the authors also use a two-pass connected component labeling algorithm to identify the separate foreground objects and their bounding boxes. To track each object, a multiple hypotesis tracking method takes place where the size and position (predicted by a Kalman Filter) of each blob is probabilistically matched with the existing tracks. Unmatched blobs in consequent frames are assigned a new track with the corresponding Kalman Filter and unmatched tracks variance is increased until their fitness (the inverse of the Kalman Filter's variance) is too low and therefore they are forgotten.

Authors state that the biggest issue with their approach is when shadows of moving objects are observed and multiple objects overlap (due to the simplicity of the multiple hypothesis tracker used).

Tracking vehicles and pedestrians in unstructured scenarios

The authors in [20] exploit the synergy between Temporal Differencing of frames and Image Template Matching, with the goal of tracking and classifying humans and vehicles in streams of unstructured outdoors scenes, in a similar fashion to the work done in [25].

The main difference between these two approaches lies in how the tracking and classification is done. In order to more robustly track each object, the authors first classify each blob (note that in [25] the authors classify last). The idea is that it is easier to match each track to the corresponding object

if the information regarding which class it belongs to is available. Classification is done based on the dispersedness of each blob. Dispersedness can be seen as the complexity of the shape in relation to its area (humans being more complex in shape than vehicles).

$$Dispersedness = \frac{Perimeter^2}{Area} \tag{2.10}$$

However, since noise or other objects that are of no interest can be detected the final classification only takes place once a blob has persisted in time long enough to be deemed relevant. Persisted in time means that the same blob can be matched to itself, by being the closest to its last location in consecutive frames and being classified by the same class. This step greatly improves the reduction of false positives in tracking, caused by temporary movement in the background, such as waving trees in the wind.

Instead of classifying based on a single observation (at each frame, independently) an histogram of the classification for the last n frames is kept and the peak is the class assigned (a simple form of maximum likelihood estimation). This helps when an object is briefly misclassified since the history in n compensates for it. Once every blob has been classified, they are matched with the existing tracks by means of image template matching (correlation between image templates). Each track has an associated image template reference (R) which is updated with an Infinite Impulse Response filter at each time-step t, according to the current blob image M which is obtained from the detection step.

$$R_t = \alpha * M_t + (1 - \alpha) * R_{t-1} \tag{2.11}$$

The way tracking is achieved means that no estimation of positions takes place (such as the use of Kalman Filters, reviewed before), making the system more robust to objects that move in an unpredictable fashion and less computationally expensive.

Real time classification for limited-data scenarios

A classification approach for limited-data scenarios is proposed in [23]. The authors define a limited-data classification scenario as the classification of a given image based on a small dataset. The classification method is based on the concept of linear regression. This approach, unlike many of the methods regularly used in similar situations, requires no training or feature extraction.

This method consists in rebuilding a given image that we want to classify on the subspace of the collection of images belonging to each class. When rebuilding an image purely with a linear combination of other images, there will be a difference between the original and the projected image on the subspace. This is called the residual value. This value allows us to compare how well each class image set is able to rebuild the original image, and thus classify it. Furthermore, this method allows us to project multiple images at once, using the regression in its matrix form. This is useful when we have multiple images of

the object we want to classify.

The authors propose multiple methods on how to classify an image set based on the residual values of each individual image. Consider the set of residuals R, where R_i^j is the residual value of the i^{th} image projected onto the j^{th} class. The first proposed classification method is based on the majority voting (MV) concept. It classifies an image set by considering the mode of the lowest residual class for each image:

$$mode(argmin_j(R_i^j) \forall i \in R) \tag{2.12}$$

If there is a tie between classes, the class with the minimum mean residual for that image set is chosen. The second proposed classification method is based on the nearest neighbour concept. After computing the minimum residual for each class j:

$$min(R_i^j) \forall i \in R) \tag{2.13}$$

the nearest neighbour methods picks the class with the lowest minimum residual value as the final classification. The last method first attributes a residual weight sum to each class. Consider that each residual value in R is transformed into a weight as follows:

$$\omega_i^j = e^{-\beta R_i^j} \tag{2.14}$$

Then, the method known as Exponential Weighted Voting choses the class that has the lowest summed weight as the final classification:

$$argmax_j(\sum_i \omega_i^j) \tag{2.15}$$

2.3 Analysis

First, it is important to point out that only two of the reviewed related work, [28] and [25], are directly comparable with the work to be done in this thesis. Other approaches reviewed in the last section share the abstract problem, whether in the domain of object detection, tracking or classification. Furthermore, the major constraint of having to run in real time means that some of these approaches cannot be applied for the problem at hand. Regardless of our specific constraints, there are some ideas that we can borrow from each of them.

2.3.1 Detection

The most common technique to use for object detection is some form of Background Subtraction, as seen in section 2.2, since it allows focusing the tracking/classification methods in regions of the frame that are more likely to be on target (belonging to an object of interest), instead of having to run them on

the whole of each frame. Which not only introduces potential false-positives (due to background regions with similar features to those of the target objects) but also requires more processing power.

Background Subtraction consists in having a model of the background and then, by comparing it to the new image, selecting regions that differ from it. This model can be either a statistical description (in [25], [15] and [26]) or an image reference itself (in [25], [20]). In the case of a statistical description the comparison checks whether the values of the new image fit the description or not. This approach is robust since it allows distinct intervals of values to be considered (applied through the use of Adaptive Gaussian Mixture Models, for example) entailing that problems such as the detection of background repetitive motions are avoided. However, objects that move slowly represent a problem since they pollute the model with noise. While this is not usually a problem for the simpler Background Reference Image approach, misclassifying regions where an object has just been identified as foreground is, due to changes in values within those regions. It works by directly comparing the new image with the background reference and checking if the values are similar, such as in [20] and [25]. This is the reason why [25] uses both types of approaches at the same time. When combined together, they offset each others shortcomings and produce a better output than any of them alone.

In other approaches, as seen in [28], detection takes place by dividing each frame in multiple regions of variable resolution and checking how likely an object is to be present by comparing its features against the ones found by either CNN or SVM that most accurately describes the training dataset. However this means that a larger area has to go through the more demanding algorithms which means that despite the usual better results, this approach is rarely applicable for real-time applications. Another common issue is that neighboring regions could be detecting only parts of an object, or, two similar objects might be in adjacent regions and thus the merging and splitting of regions has to be taken into consideration, making it harder to get accurate bounding boxes for objects.

2.3.2 Tracking

Tracking objects is highly dependent on the correct segmentation of motion done by object detection. It is a problem of data association. More specifically it associates each detected object with its past location history. The tracking of multiple objects is most commonly done through Feature Matching, i.e., each track has an associated number of features, which are compared to the features of each detected object and then matched to the object which most closely matches the overall features (called a global similarity function).

In [25] an object observation is assigned to an existing track if the value of feature similarity functions (Blob Shape Features or Color Histogram) are above a predetermined threshold. While some accuracy might be lost because no position is predicted it should still perform well in situations where the objects are not moving too fast as to cross paths in a single time-step, and possibly perform even better in

situations where the movement is unpredictable, as in our case.

In [20] each track has an associated image template that is updated as new observations for the associated object are made. An observation is associated to a track if the correlation between the image template and the observation is high enough. This approach seems adequate when objects are distinct in appearance. However it is much more prone to drifting if the detection step is not performing well enough.

The method used in [26], a linear predictive multiple hypothesis tracking, assumes that the movement of each object is linear (constant velocity, no acceleration). This means that while results are much better when the assumption is true it can perform worse than other simpler methods when it is not.

A variation of a multiple hypothesis tracking (named A Priori Assignment by the authors) used in [15] makes the previous assumption too. However, it also takes into account a color description of the object making it more robust. If a more suitable prediction method is used it could be a good approach for my case. The authors even implement the merging and splitting of tracks to handle objects overlapping, which is one of the biggest issues that seem to occur in fish tracking (as they regularly swim along/past each other).

2.3.3 Classification

From all reviewed research, the only one actually using formal classification algorithms is [28]. All others mostly leverage the performance of their detection and tracking. This allows the classification to be simple yet effective since the regions to be classified are known to be objects. And thus, only requiring to be distinguished among themselves, which seems more inline with the constraint of real-time support.

In [28], a support vector machine approach is used to classify the features extracted by an HOG of a given region. While the authors concluded that the other approach (analyzed next) provides better results, they did not seem to take into consideration that an SVM (or most classification algorithms, for that matter) are only as good as the features/metrics used. The fact that only an HOG was used seems inadequate when there are other features that can be easily extracted such as the color histogram (seen in multiple other similar works, with good results). However, since the SVM was used as a baseline I can see why they would not want to spend too long looking for better features/metrics.

The main subject and focus of their work was a CNN. While it provided relatively good results without any form of guidance from other techniques I consider it to be inadequate for my problem since it not only requires a relatively big dataset of examples for each species but is also too computationally heavy to run in real-time. Also it is worthy to note that the authors state that the CNN is not very robust to background confusion and that fish overlapping or being partially occluded is an issue. Both things are relevant for my own work and have to be taken into account for decent results.

The classification seen in [26] is an extreme example of relying on good motion segmentation/tracking

methods and taking advantage of the temporal aspect. The authors simply classify each object based on the average ratio between width/height over a time interval, achieving good results. This means that, while some accuracy might be lost, the real time performance becomes easily achievable while leaving more room to improve detection and tracking.

In [15] authors take advantage of the fact that the target object classes (pedestrians and vehicles) change their shapes as they move (pedestrians constantly change their shape by moving their legs while vehicles remain relatively constant in shape). Once again this is only possible when taking the temporal aspect into account (as a single observation would mean no access to this information). The classifier is also a simple binary threshold of the Recurrent Motion Ratio.

The method used in [20] is similar to the previous one. The authors use the complexity of an objects appearance perimeter (dispersedness). A multiple hypothesis approach is used to take into account the temporal aspect. Objects that are assumed to be the same over consecutive frames (based on proximity) are only classified as a certain class if the dispersedness remains in the bin of the same class in both instances. This is another example of exploiting the temporal aspect while keeping the classification metric simple yet effective.

One important issue to note, however, is that most of the reviewed cases are classifying two distinct classes of objects (binary classification) while in our work it will require intra-class classification (multiple different species of fish).

The method proposed in [23] is able to work with a limited dataset and requires no training. This is an advantage when building a large dataset is not feasible. Furthermore, it performs well on low resolution data which is usually the case when working in real time video, where network bandwidth or processing power might be a problem.

3

Implementation

Contents

3.1 System Architecture

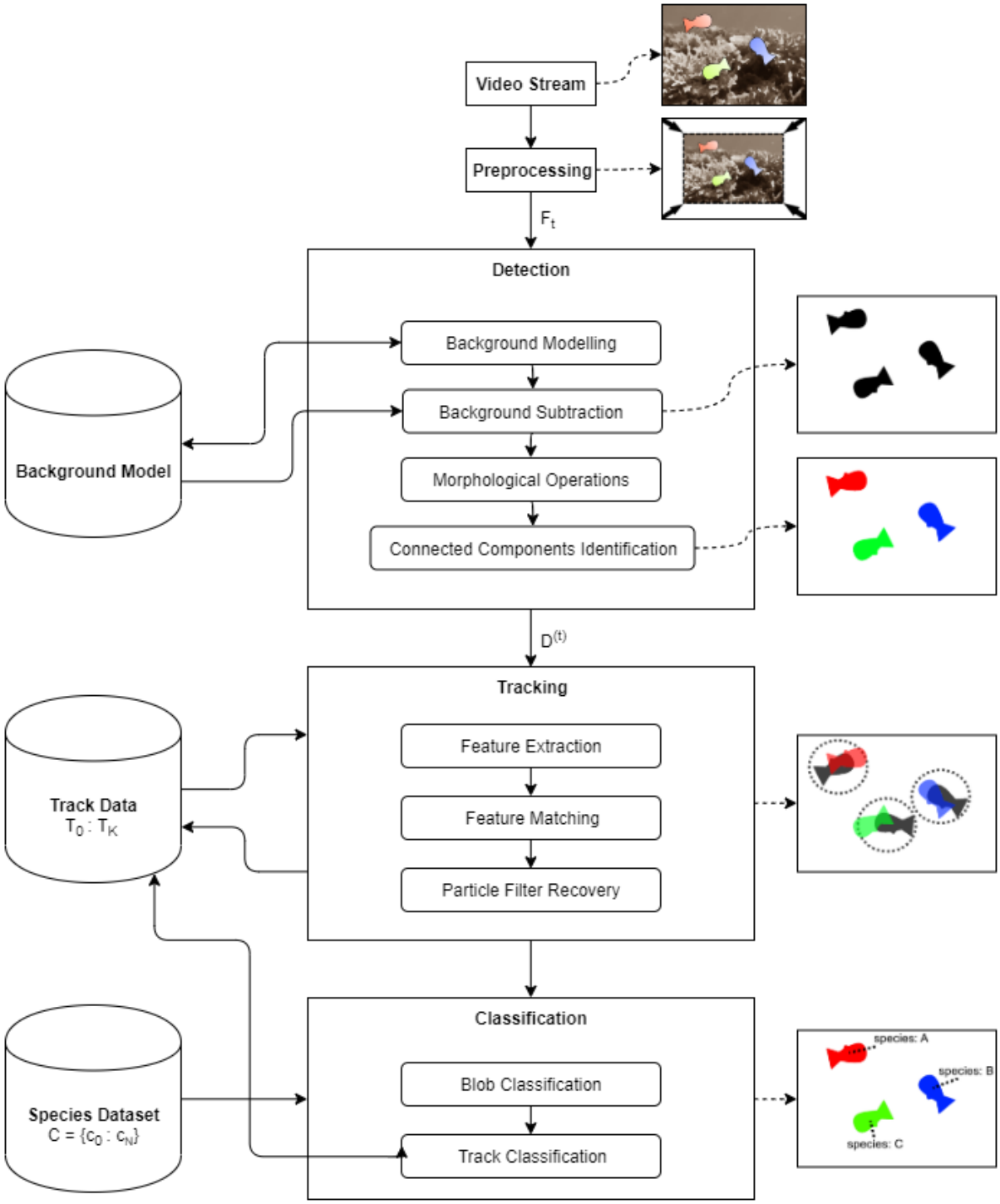


Figure 3.1: System Architecture

Our system is composed of three main modules, as depicted in figure 3.1, which frames the pipeline of

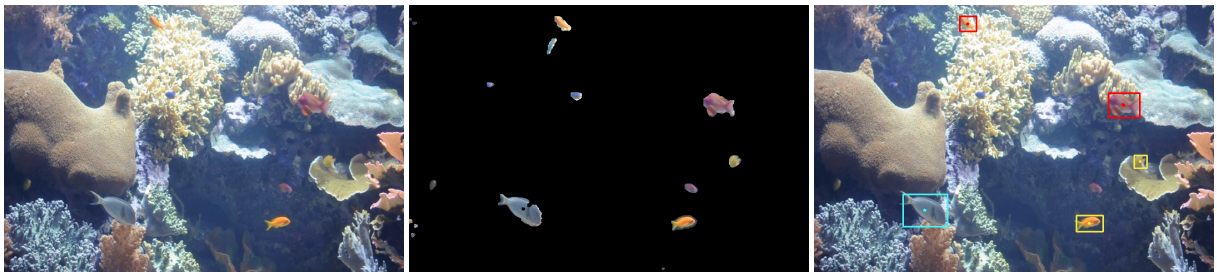
our solution. Each module has its workings discussed further, in their respective sections of this chapter.

The three modules work in sequence for each frame of a video sequence, after being pre-processed into the required format. Each frame is simply a picture output of the camera, F_t, depicting the scene at a given time-step t, as seen in figure 3.2(a).

For each frame, F_t, the Detection Module identifies which parts correspond to objects of interest by subtracting the background from the frame. This is done by keeping an updated model of the background and then using a background subtraction algorithm. Each separate foreground region is then processed into a blob. A blob contains the color values for a given region that an object lies on, as well as the shape and position of that object. Figure 3.2(b) showcases the blobs detected on a frame.

These blobs are then associated into tracks by the Tracking Module. Each track is a set of blobs corresponding to the same object for sequential time-steps. Blobs are associated to the correct track via color and position similarity analysis. When a blob cannot be associated to an existing track, a new one is created. Naturally, if a track is not active for long periods of time, it is deleted. A few active tracks are seen in figure 3.2(c), where their bounding boxes for that frame are identified by the colored rectangles.

Finally, the classification module attempts to recognize the species of each tracked fish. It achieves this by comparing the tracked objects to the second system-wide input of our system: the species dataset. This is a collection of sample pictures for all species present in a scenario. We use a closed world assumption, meaning that the present species do not change over the course of the observation. Moreover, we assume that all individuals that do not belong to a given identified species will be classified into a ”waste basket” species group. These two assumptions allow us to somewhat simplify the classification problem but are perfectly valid and true for our scenarios. Once this step is complete the cycle starts once again, continuously providing the location and species of fish within a video. In figure 3.2(c) the active tracks are color coded by class, corresponding to the different species of fish.



(a) Raw Frame **(b)** Detected Blobs **(c)** Bounding Boxes and Classes

Figure 3.2: Pipeline cycle over a video frame

3.2 Pre-processing

The pre-processing stage of our system ensures that the video data is transformed into the format that is expected by the remaining modules. First, a frame is resized into a resolution of 720 by 480 pixels. Then it transforms the colors values into the RGB format, if they are not in that format already. Both these operations are done through the respective *opencv* [6] functions. A timestamp (a value representing the instant in time) is also attached to the frame for future reference.

The output of the pre-processing is a timestamp and frame pair:

$$(t, F_t) \tag{3.1}$$

where t is the timestamp and F_t the frame matrix such that $F_t(x, y)$ is the RGB value of the x^{th} left-most and y^{th} top-most pixel, as shown in figure 3.3.

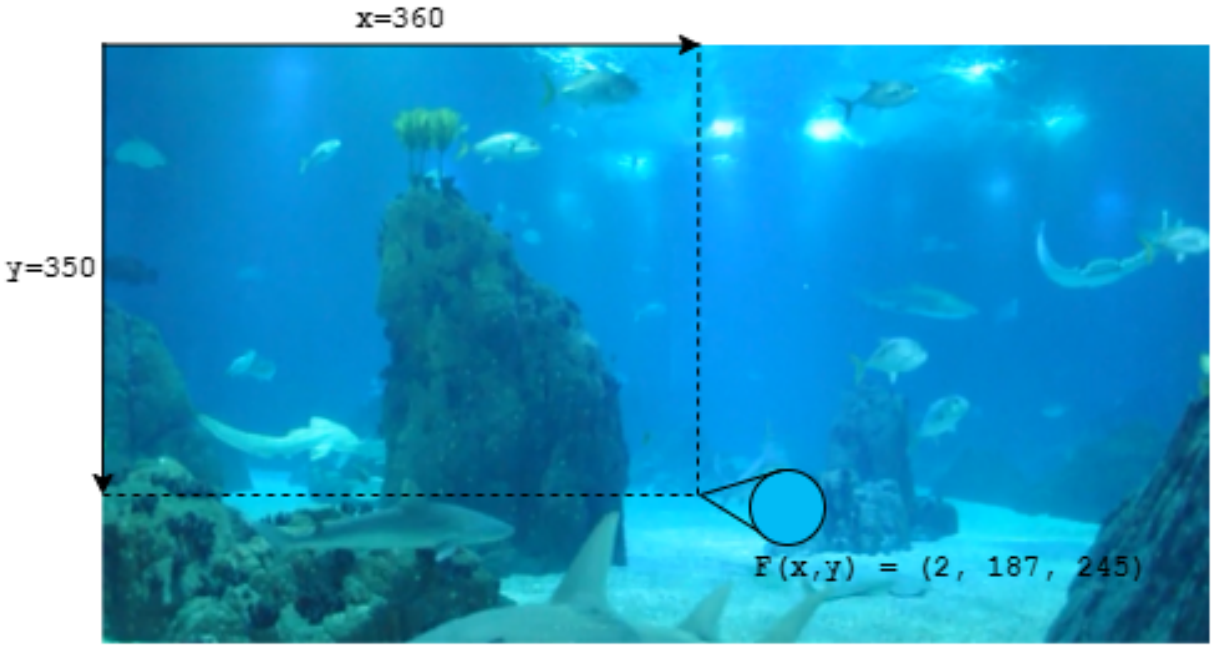


Figure 3.3: Example frame, F_t, for a given video sequence.

3.3 Detection

The detection module is responsible for localizing the fish present within a video frame, $D^{(t)}$. To do this it must first be able to recognize what is or is not part of a fish. As seen in the background section (2.1) this is commonly achieved using Background Modelling followed by Background Subtraction techniques. An Adaptive Gaussian Mixture Model (AGMM) is used for modelling the background of our scenarios. It

is a per-pixel model that identifies which color values are likely to belong to the background. This model might not be accurate at first, but it stabilizes as it updates with the data observed in more frames. Moreover, it is able to account for recurrent motion of background entities, such as coral and algae flailing in the water currents.

Our system uses the AGMM implementation provided by *opencv*, detailed in [30]. It has a few key parameters that dictate its behaviour. The first parameter is N_f, that controls how many of the past frames are considered when updating the model. N_m is the maximum amount of Gaussian components considered for each pixel. A number of variance variables dictate the initial, maximum and minimum variance values of the Gaussian mixtures. These have been carefully studied by the authors and have well established canonical values. Finally, the most important parameter is the learning rate, α, which dictates the adaption rate of the model and depends on the dynamics of the environment. When processing a frame, each pixel is considered independently during the background modeling and subtraction stages.

3.3.1 Background Modelling

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, $\vec{x}^{(t)}$, at timestep t: $X_T = \vec{x}^{(t)}, ..., \vec{x}^{(t-T)}$. 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:

$$\hat{p}(\vec{x}|X_T, BG + FG) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\hat{\vec{\mu}}_m, \hat{\sigma}_m^2 I) \tag{3.2}$$

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

$$\hat{\pi}_m \longleftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) \tag{3.3}$$

$$\hat{\mu}_m \longleftarrow \hat{\mu}_m + o_m^{(t)}(\alpha/\hat{\pi}_m)\vec{\delta}_m \tag{3.4}$$

$$\hat{\sigma}_m^2 \longleftarrow \hat{\sigma}_m^2 + o_m^{(t)}(\alpha/\hat{\pi}_m)(\vec{\delta}_m^T\vec{\delta}_m - \hat{\sigma}_m^2) \tag{3.5}$$

where $\vec{\delta}_m = \vec{x}^{(t)} - \hat{\vec{\mu}}_m$ 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 [8], and 0 otherwise. In the case where there are no close components, a new one is generated according to the previously mentioned initial parameters. If the amount of components exceeds the limit, the least important (lowest weight π_m) component is discarded.

3.3.2 Background Subtraction

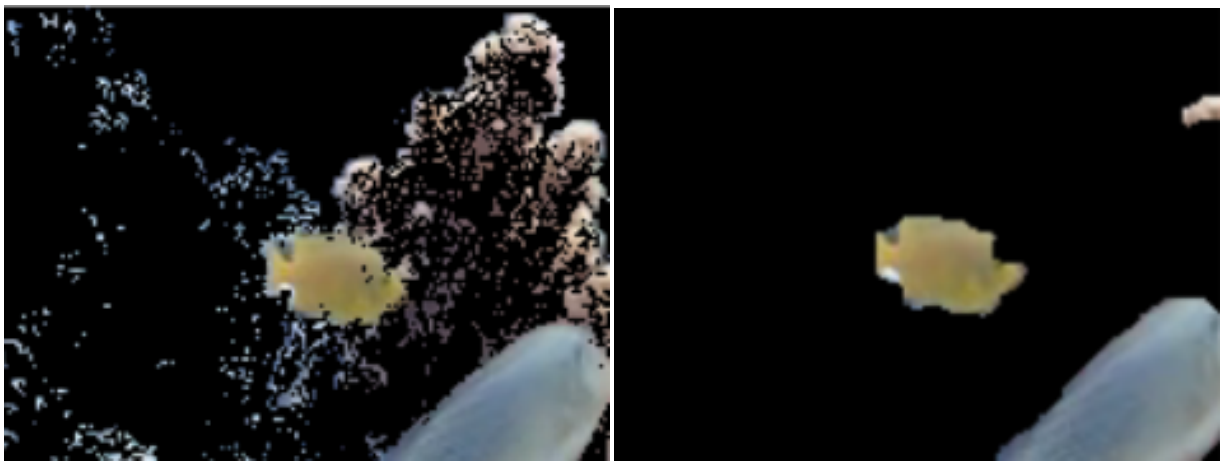
During this stage a background subtraction algorithm identifies which pixels belong to the foreground. This is done by first considering the N most important Gaussian components for each pixel. N is determined as:

$$N = argmin_n(\frac{\sum_{i=0}^{n} \hat{\pi}_i}{\sum_{i=0}^{N_m} \hat{\pi}_i} >= 0.75) \tag{3.6}$$

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 [8] between a Gaussian component m with mean value $\vec{\mu}$ and an observed value $\vec{x}$ is:

$$D_M(\vec{\mu}, \vec{x}) = \sqrt{(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T} \tag{3.7}$$

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 figure 3.4(a).



(a) Raw foreground **(b)** Filtered foreground

Figure 3.4: Applying morphological filters to the pixels flagged as foreground.

3.3.3 Morphological Operations and Connected Components

The foreground (fig 3.4(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 figure 3.4(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. These regions are the blobs seen in figure 3.2(b).

3.4 Tracking

Tracking an object consists in knowing its position at each instant. The detection module provides all detected objects in the form of blobs, with their respective positions and associated data (shape and color). With this information, we are able to associate each detected blob with an existing track or create a new track when deemed appropriate. To check if a blob should be associated with a given track, two similarity measures are taken into account. First, a predicted position for each track, based on its previous known positions, is compared to the positions of the blobs. Only the closest blobs to the predicted position of each track are considered. Then, each of the considered blobs is compared to the tracked fish in terms of color similarity. Using the Bhattacharyya coefficient, the overlap of color histograms between the last blob of a track and the detected blobs, the most similar detected blob is chosen. A minimum color similarity threshold avoids associating an unrelated blob to a given track. When a track has not been associated with a blob for a given frame, we make a second effort to find the location of its fish with a similar method to the one used in order to detect the fish in the first place. We compare the color values of the pixels in the region surrounding the predicted position of a track with the histogram its most recent blob. Finally, blobs that were not associated with any existing track are inserted into new tracks.

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.

3.4.1 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:

$$\mathcal{S}_p(D_j, T_k) = 1 - \frac{\|P_K(T_k) - P(D_j)\|}{M_d} \tag{3.8}$$

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.

3.4.2 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 number 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, gives us 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:

$$B_c(H_1, H_2) = \sum_{i=0}^{n} \sqrt{H_1^i H_2^i} \tag{3.9}$$

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:

$$\mathcal{S}_c(D_j, T_k) = B_c(H(D_j), H(T_k)) \tag{3.10}$$

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.

3.4.3 Data Association

With the two similarity functions defined in section 3.4.1 and 3.4.2, based on the location and appearance of the detected fish and tracks, 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:

$$\mathcal{S}_p(D_j, T_k) \geq \theta_p - \gamma_{p,T_k} \tag{3.11}$$

$$\mathcal{S}_c(D_j, T_k) \geq \theta_c - \gamma_{c,T_k} \tag{3.12}$$

$$k = \mathrm{argmax_i}(\mathcal{S}_c(D_j, T_i)) \tag{3.13}$$

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.

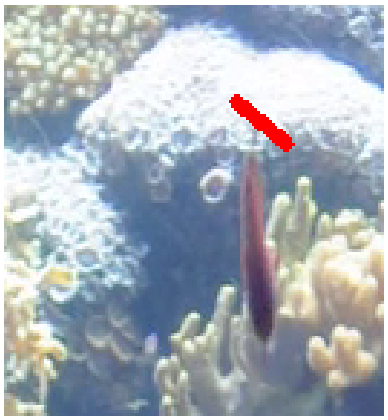


Figure 3.5: Predicted path of a fish by a Kalman Filter.

3.4.4 Particle Filter Recovery

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. In figure 3.5, for example, the line goes from the last position of that track to the predicted position by the Kalman Filter for that frame, but no match with a detected blob took place. This 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 a presence heat-map around the predicted position of the tracked fish. On the left side of figure 3.6 the previous location for a track is represented by the green square. The yellow square shows the predicted position for that track on the current frame, which was not matched with a detected blob. 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 use of the color histogram of past blobs for that track. Then each pixel is flagged as either positive (meaning it probably is part to the fish) or negative, through a minimum probability threshold parameter. The positive regions are highlighted in green on the right side of picture 3.6, while the negative regions are colored in red. 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. If this was possible it would sometimes cause a drift of the color histogram for that track, possibly resulting in the repeated tracking of a background region.

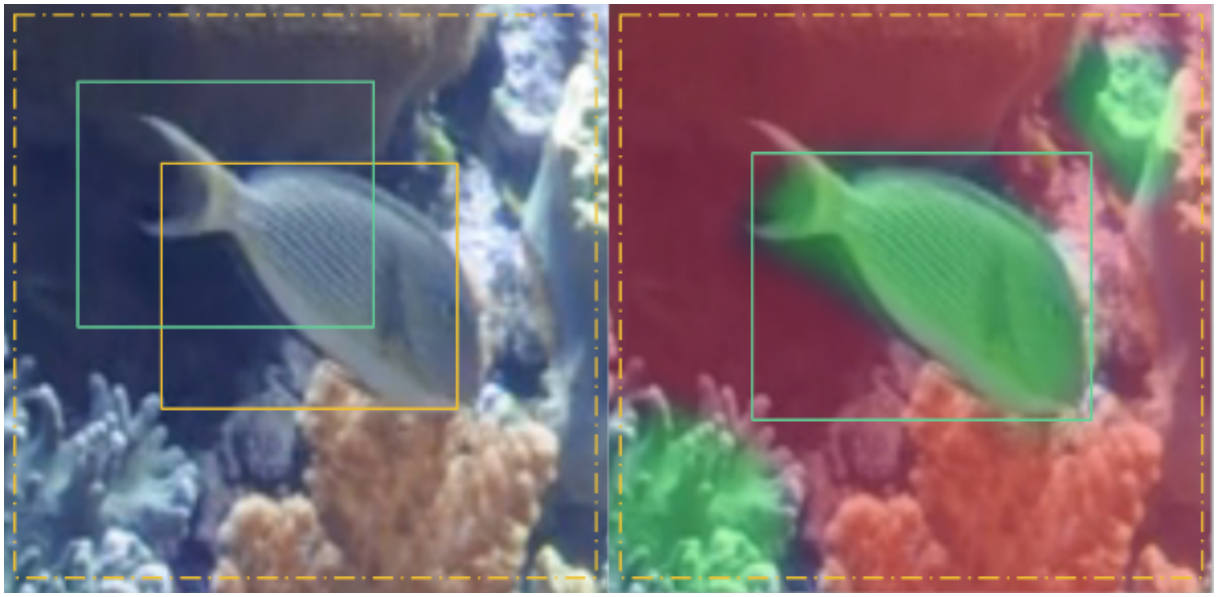


Figure 3.6: Attempting to locate the position of a tracked fish that was not matched.

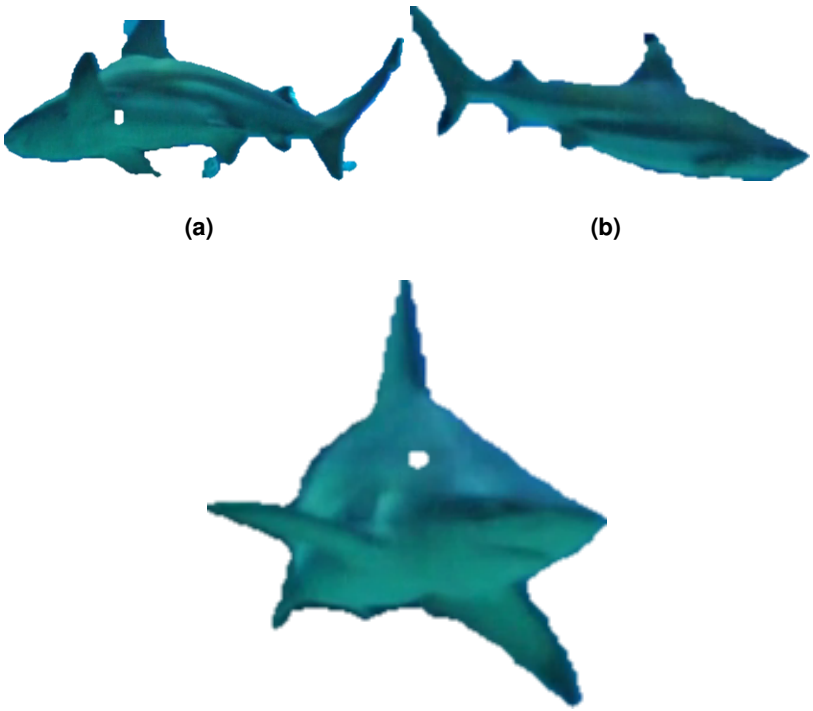
3.4.5 Track Lifespan

Detected blobs that were not matched to an existing track, through the method described in subsection 3.4.3, 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.

3.5 Classification

The last module of our system aims at identifying which species a tracked fish belongs to. The only input in this module is a collection of sample photos for each species of interest. Each species should have about 20 examples, with the background removed, as seen in figure 3.7.

By comparing characteristics of a tracked fish with the set of images of each species we should be able to identify the correct species of that fish. These characteristics change depending on the environment and species in question. Thus, the most relevant characteristics for each environment should be taken into account when possible. Take a coral reef environment, for example: the colors are well defined, the water is not murky and hence comparing the color hues between images might



(a) **(b)**

(c)

Figure 3.7: Three sample photos from our shark classification dataset.

be a feasible approach. However this is not always the case. In our test cases we faced two different environments, with distinct characteristics, that we handled differently.

3.5.1 Coral Reef Scenario

One of the environments is a coral reef. As stated before, 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 the Bhattacharyya coefficient. This method is not as reliable when lighting drastically changes the colors of fish, but is able to handle our test scenarios successfully.

3.5.1.A Features

Consider again 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. Once again, if H_1 and H_2

have n bins then their coefficient is computed as follows:

$$B_c(H_1, H_2) = \sum_{i=0}^{n} \sqrt{H_1^i H_2^i} \tag{3.14}$$

3.5.1.B Classification

In order to classify a track we consider the average Bhattacharrya coefficients, B_c, of the last $k <= 5$ 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 for that timestep. 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.

3.5.2 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. The main contributing factors for this is the depth of the water and the fact that it is an enormous tank lit only by natural light coming from the top of the building. The sheer size of this tank has two consequences that make our task harder. First, due to its 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, with less detail than ideal. To overcome these two problems a solution that does not rely on color hues or the scale of the fish is required. An histogram of oriented gradients (HOG) fits these requirements, when treating images scaled to the same dimensions.

3.5.2.A 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.

3.5.2.B 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. A good analogy for this would be attempting to reconstruct an image of a goldfish with pieces from an

unrelated shark puzzle, and then the pieces of another goldfish puzzle. The other goldfish puzzle would probably come much closer to accurately reconstructing the original image. Then, the residuals, which are the differences between the original and the rebuilt images, dictate which species more accurately rebuilt the image. By down-scaling images we are able to keep this method running in real time. In the following paragraphs we describe the process mathematically. This linear regression approach is based on the work in [23], where a more in-depth description can be found.

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:

$$\gamma_i = \zeta_j \theta_j \tag{3.15}$$

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

$$\theta_j = (\zeta_j^T \zeta_j)^{-1} \zeta_j^T \gamma_i \tag{3.16}$$

$$\hat{\gamma_j} = \zeta_j \theta_j \tag{3.17}$$

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

$$R_i^j = ||\gamma_i - \hat{\gamma_i}|| \tag{3.18}$$

A smaller residual value means the rebuilt feature vector is closer to the original than otherwise. This principle can be scaled into matrix form, computing the residuals to more than one image at once. This allows us to take into account the last $n <= 5$ blobs associated with a track, improving the results of our classification. 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:

$$\omega_i^j = e^{-\beta R_i^j} \tag{3.19}$$

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

$$c = argmax_j(\sum_i \omega_i^j) \tag{3.20}$$

3.5.3 Temporal Consistency

Regardless of which of the two methods used, to further improve the stability of our classification module we take advantage of the fact that a fish does not change species. In other words, as long as the tracking is working properly, a fish species should remain constant. By taking the most frequent classification of a track in the previous $n = 20$ time steps we allow our classifier to have a margin of error. This means that sporadic image classification errors should not impact the overall track classification. In figure 3.8, the result of a processed frame is shown, where each bounding box is color coded by class.

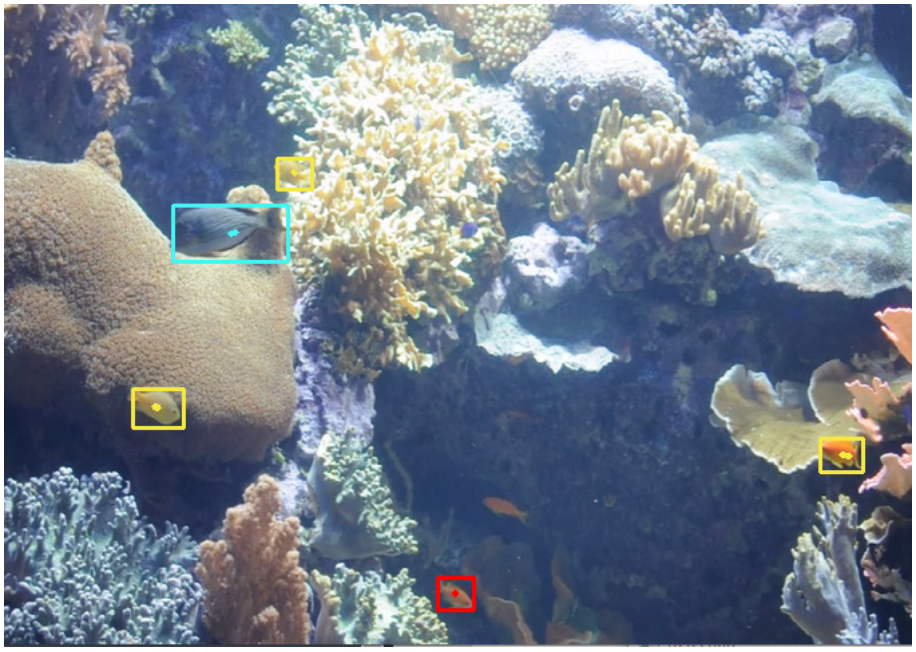


Figure 3.8: Tracking bounding boxes, color coded by class.

4

Experiments

Contents

4.1 Setup and Data-set

In order to develop and test our system, and in collaboration with the *Oceanário de Lisboa*, we recorded videos from different environments in different tanks and settings. We captured videos in different settings since different environment properties pose distinct challenges.

Our setup when recording these environments consisted of a camera (Nikon d5300) on a static support, recording a given environment for up to five minutes at a time. The tanks were filmed from the outside, meaning a thick glass barrier may have visually distorted the scene in some cases. From the recorded videos we selected two five minute segments which have the most of the action. This helps us test our system, exposing it to as many interactions and edge cases as possible, as well as testing the amount of fish it can handle simultaneously. We also recorded videos with a GoPro camera in the same settings. However, the GoPro has a wide angle lens that limits the distance at which the fish are of adequate size in the frame for our system. For this reason we did not use the GoPro footage to develop and test our system.

The first scenario takes place in the main tank of the *Oceanário de Lisboa*, as seen in figure 4.1(a). This tank mimics the open ocean waters, with a range of species from different oceans that would not normally be seen together. The presence of fish schools is also a challenge. This scenario mainly allows us to determine the maximum depth of water that our system is able to process, and how it might affect the performance of our methods, due to the impact it has in the observed color pallet. The main challenges we identified in this environment were:

- the environment and fish color uniformity, as both have a blue tint;
- the large number of fish, with some schools;
- the potential distance from camera, both from the camera to the tank and to the fish in the water;
- the presence of scene-object occlusions, specially with the rocks in the tank;
- the number of existing species, many from several distinct oceans;

The second scenario was filmed in a secondary tank in the *Oceanário de Lisboa*, seen in figure 4.1(b). This tank houses a smaller coral reef. In this scenario the colors are vibrant, the fish are more consistent in size and the water is very clear. While this tank contains a smaller set of species, it provides its own unique set of challenges not seen in the first environment. We identified the following:

- the amount of background movement, mainly of coral and algae flowing with the current;
- the smaller size of most fish;
- the speed and inconsistency of fish movement;

For each scenario we created three distinct datasets that serve different purposes. The first dataset is the species example images, that were used as an input for our classification module. An example of this dataset can be seen in figure 4.2, where some of the images of our surgeon-fish dataset can be seen. Each considered class has 20 example pictures, as varied in pose and lighting as possible, where the background was removed.

The second dataset is a collection of tracked fish, with the species manually labeled. This dataset is used to test our classification module. It was built by exporting the bounding boxes of the tracks output by our tracking module in either scenario. Then, we filtered some of the tracks that had multiple species or other extreme defects, such as incomplete segmentation of the fish or multiple species in a track. This was done to isolate the classification module, and thus have the ability to test it unaffected by the performance of the other modules. Figure 4.3 shows a few consecutive frames of a track in this dataset.

The last dataset consists of the positions of different fish at each instant for a given video segment, allowing us to test the detection and tracking modules. Building these datasets was very time consuming, and in the interest of time we had to limit their extent. This, unfortunately, limits the metrics we are able to measure on our tests, since we do not have the position of all the fish present in every frame. Thus, we are unable to identify false positive detections or tracks. Tables 4.1, 4.2 and 4.3 contain descriptions of our datasets, with the amount of data gathered for each environment.

Table 4.1: Tracking Ground-truth Dataset

Environment	Tracked Fish	Total Position Samples
(1) Main Tank	18	828
(2) Coral Reef Tank	15	2245

Table 4.2: Classification Input Dataset

Environment	Classes	Images per Class
(1) Main Tank	4	20
(2) Coral Reef Tank	4	20

Table 4.3: Classification Testing Dataset

Environment	Class	Tracks	Total Images
(1)	ray	31	235
(1)	shark	49	445
(1)	striped	29	240
(1)	tuna	139	1141
(2)	angelfish	26	314
(2)	goldfish	62	873
(2)	surgeonfish	30	436
(2)	yellowtailed	46	566

4.2 Experiments and Metrics

In this section we discuss the testing methods for our system. Note that the detection and tracking modules are tested as one, since testing the detection module alone would require a more extensive dataset. Considering that the tracking relies heavily on the detection module, its performance is also a qualitative indicator of how well the detection module is working.

Each module is tested in the two discussed environments (two segments of video). This allows us to study whether the methods and their respective parameters have a significant impact on the performance of our system, as well as how they fare against the unique challenges of each environment. For the sake of consistency between the test results, tests were conducted on a single computer with the following hardware:

- **Processor**: AMD FX-6350 (6 cores @ 3.9GHz)
- **Memory**: 8GB (DDR3 @ 1866MHz)
- **GPU**: Sapphire Radeon HD 7950 3GB DDR5 (850MHz core clock)

4.2.1 Detection and Tracking

In order to test the detection and tracking systems consider the positions of the labeled tracks at each instant. Every five frames the positions of the active tracks are compared to those in the dataset. If a labeled position is within the region of an active track bounding box it is considered as correctly tracked for that five frame interval. This interval is a compromise between the accuracy of results and the time spent labelling the video segments manually.

Consider $T_o, ...T_n$, the set of tested tracks and the functions: $N_T(t)$, the number of correctly tracked positions for the test track t; $N_M(t)$, the number of incorrectly (missed) tracked positions; $R(t) = N_T(t)/(N_T(t) + N_M(t))$ the ratio of correctly tracked positions; and $U(t)$ the number of unique tracks tracking the positions of the tested track. Note that $[condition]$ is valued as 1 if the $condition$ is true and 0 otherwise. Based on these, the following test metrics are computed:

- Ratio of Mostly Tracked (RMT): the ratio of tracks correctly tracked at least 80% of the time.

$$\frac{1}{|T|}\sum_{t\in T}[R(t) > 0.8] \tag{4.1}$$

- Ratio of Mostly Lost (RML): the ratio of tracks correctly tracked, at most, 20% of the time.

$$\frac{1}{|T|}\sum_{t\in T}[R(t) < 0.2] \tag{4.2}$$

- Global Accuracy (GA): the overall ratio of correctly tracked labelled samples.

$$\frac{\sum_{t \in T} N_T(t)}{\sum_{t \in T} N_T(t) + N_M(t)} \tag{4.3}$$

- Identity Swaps (SWP): the average number of unique tracks per tested track.

$$\frac{1}{|T|} \sum_{t \in T} U(t) \tag{4.4}$$

We also test the impact of our tracking recovery method, inspired in particle filtering, by running the system with it both activated and toggled off, once we find the appropriate values for the tracking thresholds.

4.2.2 Classification

Using the track frames dataset, where each track has its corresponding history of blobs and manually labeled fish species, we are able to test our classification module. Both methods, the linear regression of HOG features and the Bhattacharrya coefficient comparison are tested.

During testing each track in the dataset is processed by our classification module, as it normally would in a real scenario. This means that a single track is possibly tested more than once, as its frames are grouped into sets of up to five frames. Then, the classification of each set is checked against the labels in the dataset for correctness.

Let $C = c_0, ..., c_n$ be the set of possible classes on a given test scenario, and $T = t_0, ..., t_m$ the set of tests. If $P(t_i)$ is the predicted class output by our classification method and $E(t_i)$ the expected class for that test, the computed metrics for our classification tests are:

- Accuracy:

$$\frac{1}{|T|} \sum_{t \in T} [P(t) = E(t)] \tag{4.5}$$

- Precision, p (class c_i):

$$\frac{1}{\sum_{t \in T} [P(t) = c_i]} \sum_{t \in T} [P(t) = E(t)][P(t) = c_i] \tag{4.6}$$

- Recall, r (class c_i):

$$\frac{1}{\sum_{t \in T} [E(t) = c_i]} \sum_{t \in T} [P(t) = E(t)][E(t) = c_i] \tag{4.7}$$

- F1-score (class c_i):

$$2 \frac{p(c_i) r(c_i)}{p(c_i) + r(c_i)} \tag{4.8}$$

4.3 Results

4.3.1 Detection and Tracking

The results of our detection and tracking tests, for either environment, can be seen in tables 4.4, 4.5 and 4.6. In table 4.4 we explore the θ_p parameter, which controls the minimum position similarity threshold for our tracking feature matching. The color threshold θ_c parameter is also tested in table 4.5. Finally, the results of changing the learning rate parameter in our detection module is explored in table 4.6. Within these tables you see the values of the global accuracy (GA), ratio of mostly tracked (RMT), ratio of mostly lost (RML) and the number of identity swaps per track (SWP) for each parameter value.

Table 4.4: Tracking test results: varying the minimum position similarity, θ_p, threshold.

Main Tank

θ_p	GA	RMT	RML	SWP
90%	$94,8\%$	$88,8\%$	0%	3,55
91%	$94,7\%$	$88,8\%$	0%	3,55
92%	$95,0\%$	$88,8\%$	0%	3,50
93%	$95,2\%$	$88,8\%$	0%	3,44
94%	$95,3\%$	$88,8\%$	0%	3,27
95%	$95,0\%$	$88,8\%$	0%	3,50
96%	$93,7\%$	$83,3\%$	0%	3,16
97%	$94,3\%$	$88,8\%$	0%	3,33

Reef

θ_p	GA	RMT	RML	SWP
90%	$82,4\%$	$73,3\%$	$6,6\%$	5,73
91%	$82,6\%$	$73,3\%$	$6,6\%$	5,86
92%	$82,6\%$	$73,3\%$	$6,6\%$	5,66
93%	$82,6\%$	$73,3\%$	$6,6\%$	5,86
94%	$82,5\%$	$73,3\%$	$6,6\%$	5,73
95%	$82,4\%$	$73,3\%$	$6,6\%$	5,80
96%	$82,3\%$	$73,3\%$	$6,6\%$	5,80
97%	$82,0\%$	$73,3\%$	$6,6\%$	6,33

Table 4.6: Tracking test results: varying the learning rate.

Main Tank

LR	GA	RMT	RML	SWP
$0,3\%$	$94,8\%$	$88,8\%$	0%	3,55
$0,5\%$	$93,6\%$	$88,8\%$	0%	3,22
$0,7\%$	$93,3\%$	$88,8\%$	0%	3,22
$0,9\%$	$92,7\%$	$83,3\%$	0%	3,17
$1,1\%$	$91,8\%$	$88,8\%$	0%	3,17
$1,3\%$	$88,8\%$	$72,2\%$	$5,5\%$	2,83

Coral Reef

LR	GA	RMT	RML	SWP
$0,3\%$	$82,4\%$	$73,3\%$	$6,6\%$	5,73
$0,5\%$	$81,7\%$	$73,3\%$	$6,6\%$	5,73
$0,7\%$	$82,1\%$	$73,3\%$	$6,6\%$	5,80
$0,9\%$	$81,5\%$	$73,3\%$	$6,6\%$	5,47
$1,1\%$	$81,2\%$	$73,3\%$	$6,6\%$	5,67
$1,3\%$	$81,1\%$	$66,6\%$	$6,6\%$	5,87

Table 4.5: Tracking test results: varying the minimum color similarity, θ_c threshold.

Main Tank

θ_c	GA	RMT	RML	SWP
90%	$96,1\%$	$94,4\%$	0%	3,28
91%	$96,2\%$	$94,4\%$	0%	3,22
92%	$96,2\%$	$94,4\%$	0%	3,05
93%	$95,4\%$	$88,8\%$	0%	3,11
94%	$95,4\%$	$88,8\%$	0%	3,28
95%	$95,0\%$	$83,3\%$	0%	3,28
96%	$94,3\%$	$88,8\%$	0%	3,28
97%	$94,3\%$	$83,3\%$	0%	3,78

Reef

θ_c	GA	RMT	RML	SWP
90%	$83,1\%$	$73,3\%$	$6,6\%$	5,40
91%	$83,0\%$	$73,3\%$	$6,6\%$	5,46
92%	$82,9\%$	$73,3\%$	$6,6\%$	5,73
93%	$82,7\%$	$73,3\%$	$6,6\%$	5,66
94%	$82,4\%$	$73,3\%$	$6,6\%$	5,73
95%	$81,5\%$	$73,3\%$	$6,6\%$	5,80
96%	$80,9\%$	$66,6\%$	$6,6\%$	6,13
97%	$79,7\%$	$66,6\%$	$6,6\%$	7,20

Table 4.7: Testing the impact our suggested methods of improving tracking-by-detection.

Main Tank - $\theta_p = 94\%, \theta_c = 92\%$

Position Prediction	Detection Recovery	GA	SWP	RMT	RML	FPS
ON	ON	93,15	2,72	88,8	0	10,97
ON	OFF	91,96	2,27	88,8	0	11,80
OFF	ON	92,56	2,5	72,2	0	11,02
OFF	OFF	91,87	2,22	88,8	0	11,91

Coral Reef - $\theta_p = 92\%, \theta_c = 90\%$

Position Prediction	Detection Recovery	GA	SWP	RMT	RML	FPS
ON	ON	83,50	4,33	73,3	6,7	15,09
ON	OFF	82,03	4,73	73,3	6,7	16,11
OFF	ON	83,91	4,07	73,3	6,7	15,02
OFF	OFF	82,78	4,26	73,3	6,7	16,18

4.3.2 Classification

The results of our classification tests in both scenarios can be seen in tables 4.8 and 4.9. These results are displayed per species, except for the global accuracy (GA). For each tested species we display the precision (P), recall (R) and the f-1 score (F) based on the parameter value. The tested parameters are the number of Histogram of Oriented Gradients (HOG) bins, in table 4.8, as well as the number of bins in the color histograms used for the Bhattacharyya Coefficient, in table 4.9. Additionally, the size of the HOG features is displayed in the column feature length (FLEN) of table 4.8.

(a) Main tank

(b) Coral reef

Figure 4.1: Environments: the main tank and the coral reef.

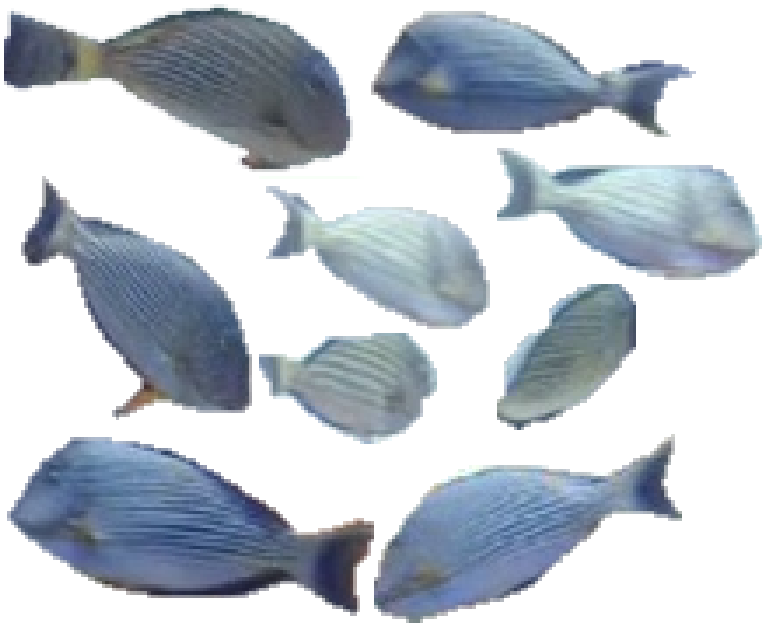


Figure 4.2: Part of the surgeon fish classification dataset.

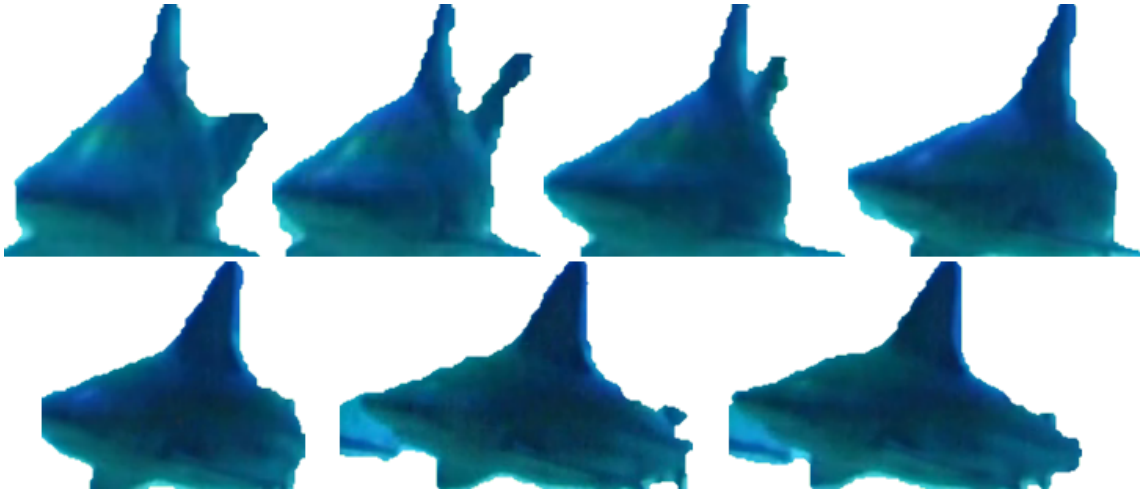
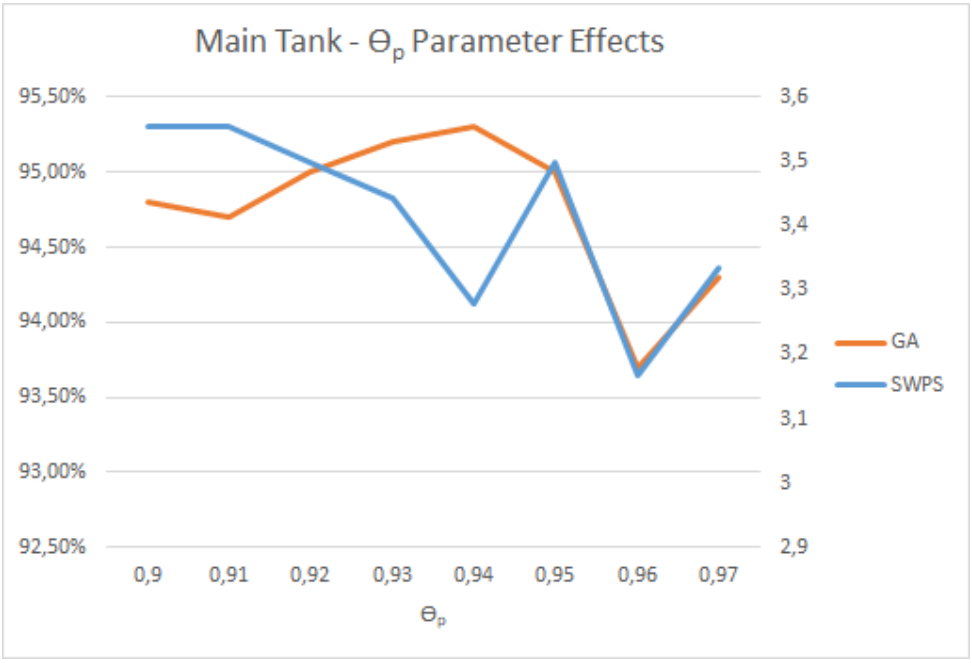
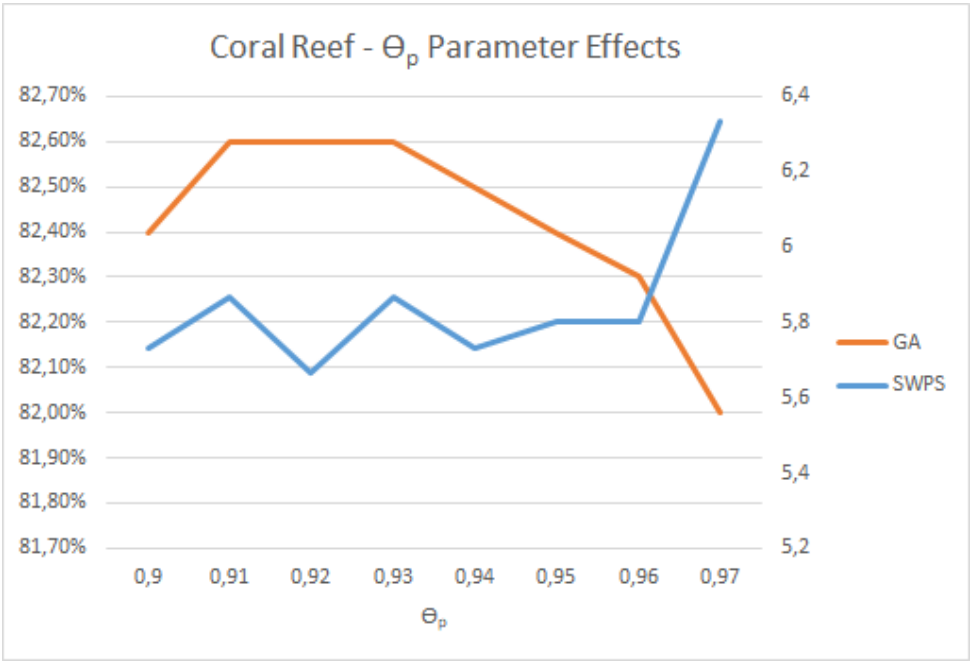


Figure 4.3: Frames of a tracked shark in our classification test dataset.

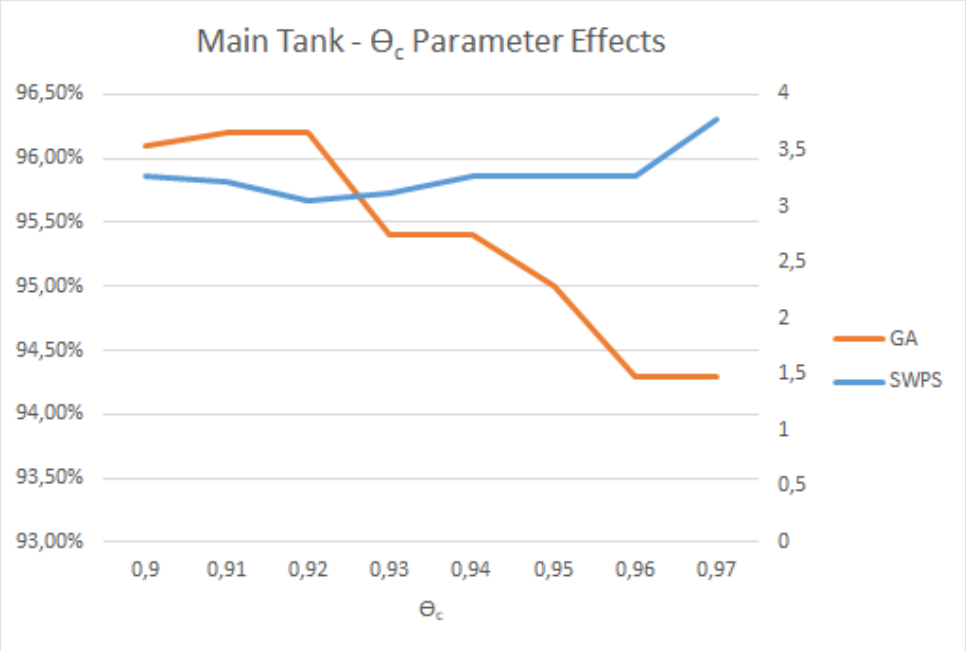


(a) Main Tank

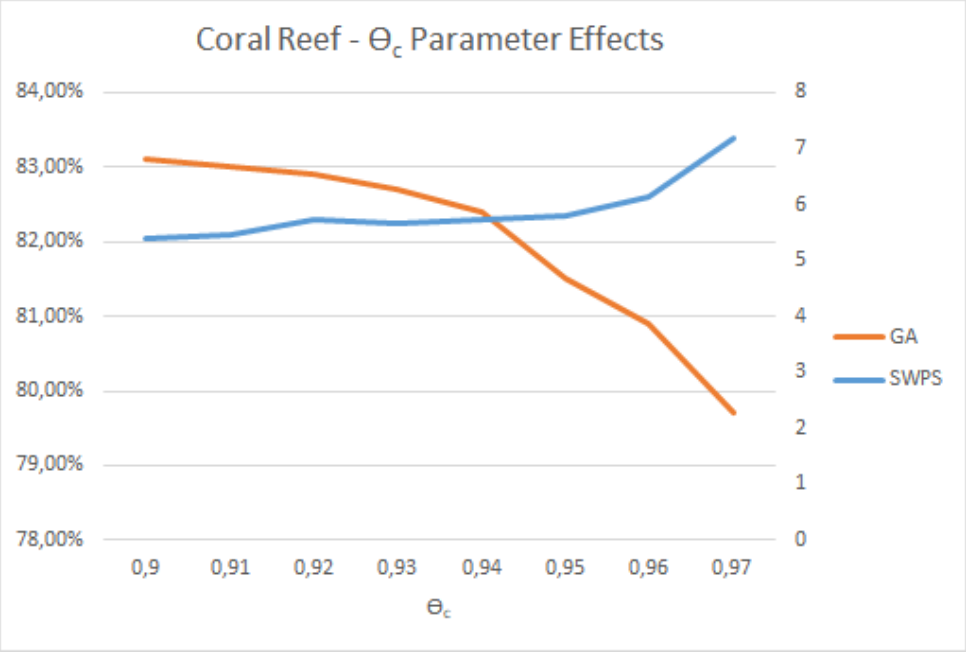


(b) Coral Reef

Figure 4.4: Plotting the effects of the minimum position similarity threshold in either environment. A higher global accuracy (GA) is better and a higher average number of identity swaps (SWPS) is worse.

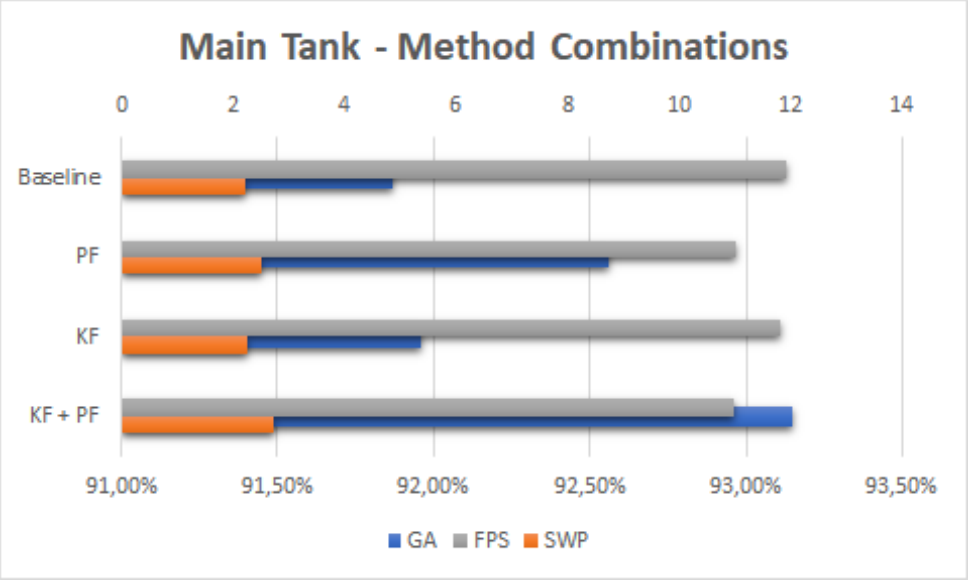


(a) Main Tank

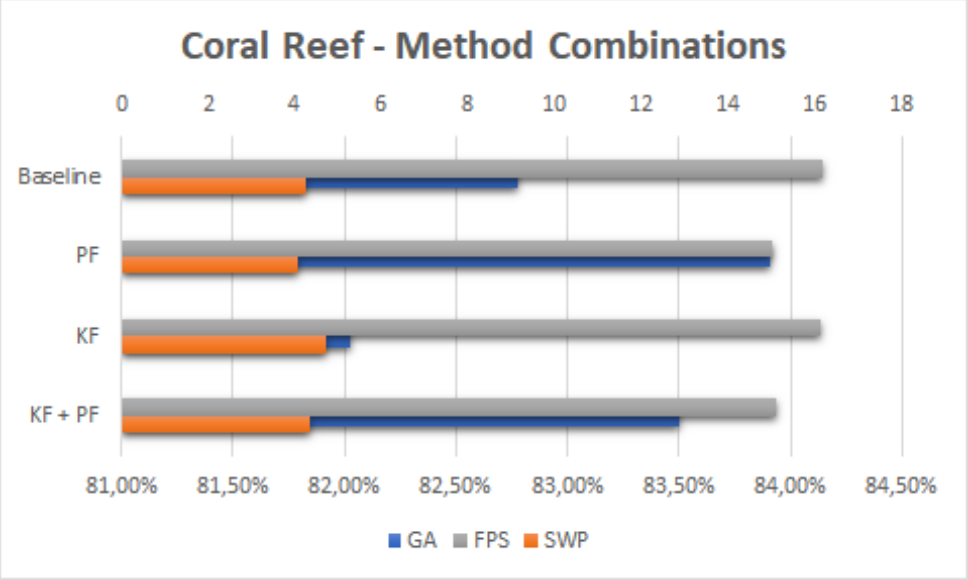


(b) Coral Reef

Figure 4.5: Plotting the effects of the minimum color similarity threshold in either environment. A higher global accuracy (GA) is better and a higher average number of identity swaps (SWPS) is worse.



(a) Main Tank



(b) Coral Reef

Figure 4.6: Plotting the impacts of the proposed improvements methods to tracking-by-detection in either environment. A higher global accuracy (GA) is better, a higher average number of identity swaps (SWPS) is worse and a higher amount of processed frames per second (FPS) is better.

Table 4.8: Main tank classification test results.

		ray			shark			striped			tuna			
HOG Bins	**FLEN**	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**GA %**
4	1936	67,6	74,2	70,8	74,1	81,6	77,7	88,9	82,8	85,7	96,5	78,4	86,5	79
5	2420	66,7	77,4	71,6	80,4	83,7	82	100	86,2	92,6	97,4	82	89,1	82,3
6	2904	69,4	80,6	74,6	80,4	83,7	82	100	86,2	92,6	96,6	82,7	89,1	83,1
7	3388	68,6	77,4	72,7	80,8	85,7	83,2	100	86,2	92,6	97,5	84,9	90,8	84,3
8	**3872**	**67,6**	**80,6**	**73,5**	**78,2**	**87,8**	**82,7**	**100**	**86,2**	**92,6**	**97,5**	**84,2**	**90,3**	**84,7**
9	4356	67,6	80,6	73,5	82,4	85,7	84	100	86,2	92,6	97,5	83,5	89,9	83,9
10	4840	69,4	80,6	74,6	84	85,7	84,8	100	86,2	92,6	96,7	84,9	90,4	84,7
11	5324	68,6	77,4	72,7	84,3	87,8	86	100	86,2	92,6	97,5	84,2	90,3	84,3
12	5808	68,6	77,4	72,7	84	85,7	84,8	100	86,2	92,6	97,5	84,2	90,3	83,9

Table 4.9: Coral reef classification test results.

	angelfish			goldfish			surgeonfish			yellowtailed			
HIST Bins	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**P %**	**R %**	**F**	**GA %**
4	86,2	96,2	90,9	98,3	91,9	95	54,7	96,7	69,9	100	52,2	68,6	82,3
5	**86,2**	**96,2**	**90,9**	**98,3**	**91,9**	**95**	**56,9**	**96,7**	**71,6**	**100**	**56,5**	**72,2**	**84,0**
6	83,3	96,2	89,3	98,2	90,3	94,1	55,1	90	68,4	92,9	56,5	70,3	81,7
7	80	92,3	85,7	98,2	88,7	93,2	53,8	93,3	68,3	96,2	54,3	69,4	80,5
8	80	92,3	85,7	98,2	88,7	93,2	55,8	96,7	70,7	100	56,5	72,2	81,7
9	77,4	92,3	84,2	98,2	87,1	92,3	54,7	96,7	69,9	100	54,3	70,4	80,5
10	78,1	96,2	86,2	98,2	87,1	92,3	54,9	93,3	69,1	96,2	54,3	69,4	80,5

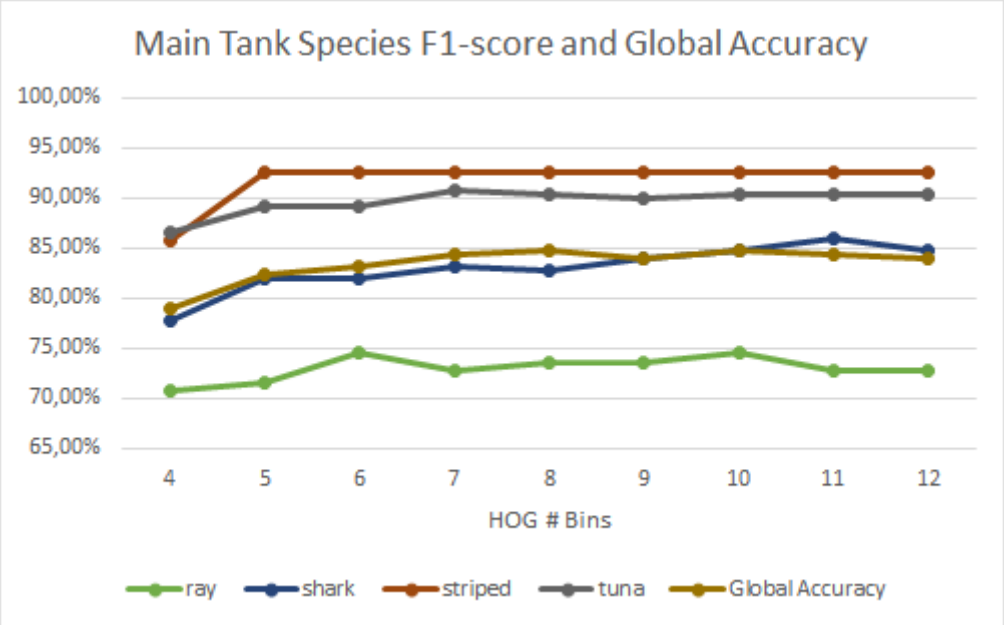


Figure 4.7: Global Accuracy and F1-Score for the species in the Main Tank scenario. Higher values are better.

4.4 Discussion

The main purpose of our tests was three fold:

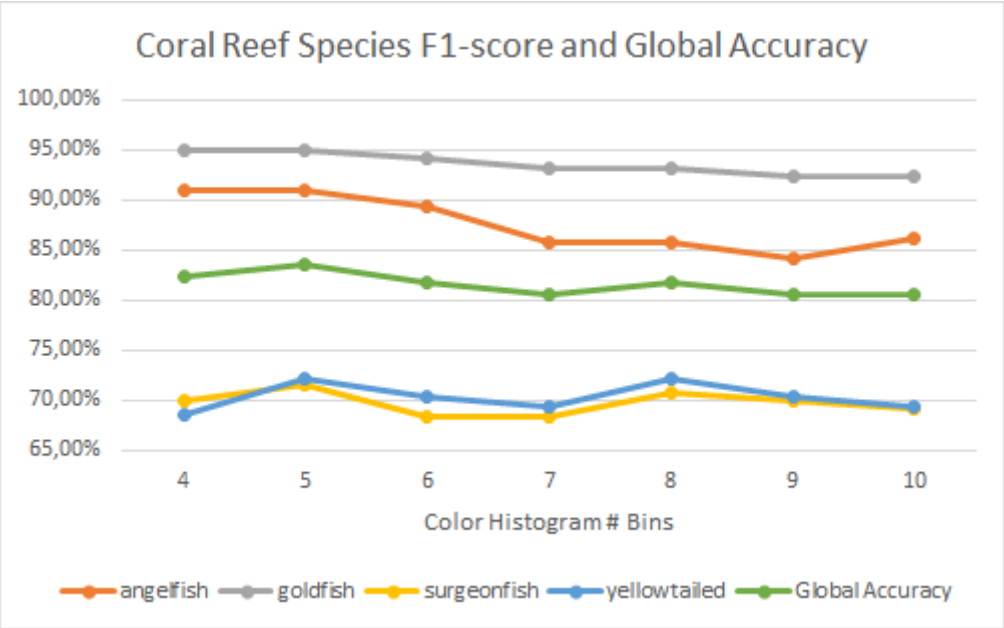


Figure 4.8: Global Accuracy and F1-Score for the species in the Coral Reef scenario. Higher values are better.

1. to find the adequate values for the parameters of the algorithms that were used;
2. to explore the impact that different environment properties have on those parameter values;
3. to evaluate the performance of our system on the test scenarios.

Detection and Tracking

The two formally tested tracking threshold parameters, θ_p and θ_c, have by far the greatest impact in the performance of our tracking module. As each test takes a considerable amount of time to run, we chose to test either parameter in increments of 1%, starting at 90%, as seen in table 4.4 and 4.5. Then, in order select the appropriate threshold values for each property (θ_p for position and θ_c for color) we first prioritized maximizing global accuracy and then minimizing the number of identity swaps. While a balance between these is important to achieve, we would require a bigger test dataset as well as more tests in order to find the appropriate weight of each metric. In table 4.10 we present the chosen threshold values, as well as the metrics obtained when tested.

As seen in table 4.10, both color and position minimum similarity thresholds are lower in the case of the coral reef scenario. The minimum position threshold discrepancy is easily explained. The movement presented by the species in the coral reef is not only faster but also less predictable. Thus, the predictions made by the Kalman filter on the expected position of a fish are less reliable. This means that we need to consider a bigger area surrounding the expected position to reliably associate the correct blob and track pairs, resulting in a lower position similarity threshold. The minimum color similarity threshold

discrepancy also has an explanation. First, the colors in the main tank are more uniform when compared to the colors in the coral reef. This results in more similar color histograms overall, and thus requires a higher threshold to reliably distinguish fish among each-other. Secondly, the number of pixels on target for the coral reef environment is lower since the fish occupy less area of a frame on average. This means that any background pixels in the detected fish blobs have not only a bigger impact but also differ much more from the colors seen in fish. This has an impact on the Bhattacharrya Coeficient between the blobs, requiring a lower similarity threshold for the coral reef.

Table 4.10: Chosen tracking parameters for each environment.

			Metrics			
	Parameter	**Value %**	**GA %**	**SWP**	**RMT %**	**RML %**
Main Tank	θ_p	94	95,3	3,44	88,8	0
	θ_c	92	96,2	3,05	94,4	0
Coral Reef	θ_p	92	82,6	5,66	73,3	6,6
	θ_c	90	83,1	5,40	73,3	6,6

We then attempted to quantify the impact of the applied methods on the performance of our system. We tested our system on those same scenarios with the multiple combinations of the applied methods enabled and disabled, as seen in table 4.7.

Taking a first look at the main tank environment we can see that, as expected, our system performs better (higher GA) 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. On its own, the particle filtering recovery method only seems to improve the global accuracy by 0,6%. When used together, however, there is a significant (1,4%) improvement on the global accuracy. Because the tracking is more consistent on a per-fish basis there are more opportunities for tracks to be swapped among two crossing fish, which could explain the higher number of identity swaps. Finally, as expected, the average number of frames processed per second is lower with these methods enabled, as they take more computation time.

On the coral reef scenario we see that while the recovery method based on particle filtering improves the global accuracy, the Kalman filter position prediction method seems to affect it negatively with and without the recovery method enabled. We believe this can be attributed to the more random nature of movement presented by the smaller fish in this environment. This also means there are bound to be more track identity swaps. 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 per-track the number of switches is 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. Lastly, as there are considerably less fish on this scenario, the system is able to run significantly faster: 5 frames per second faster than on the main tank, on average. The impact that our methods have on the frame

rate are similar to the impact observed in the main tank scenario.

Classification

The classification tests allowed us to find the adequate parameter values for either classification methods as well as the overall and per species classification performance.

The classification feature parameters: the number of bins (8) computed for the histogram of oriented gradients, in the case of the main tank, and the number of bins (5) for the histogram of colors, in the case of the coral reef, do not have an intuitive explanation for their values. We chose them based on two factors: maximizing the global accuracy and minimizing the computational complexity they entail. This means prioritizing lower values for the number of bins in both cases.

In the case of the main tank, one value stands out from the rest, the precision of the ray species is significantly lower than the rest. Rays have a unique way of moving. They have ”wings” which they 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 figure 4.9, which could explain why a classifier based on shape features presents a lower precision value for the ray species. The other classes are much less prone to this problem since their bodies deform less during locomotion and thus present a more uniform shape overall.

Figure 4.9: Part of the ray classification dataset.

On the coral reef scenario we observe two values that stand out. The yellowtailed (figure 4.10) recall (56,5%) and the surgeonfish precision (56,9%). These two values are related to the same problem. To understand it we must first look at the output of our detection module. Consider the two blobs ouput by our detection in figure 4.11. Note that the yellowtailed blob is resized, it contains significantly less pixels on target when compared to the surgeonfish due to the size of these fish. Additionally, 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 expect color histogram. Although it might be easy for us to distinguish

between these two, specially due to their size, they are quite similar in color under certain lighting. Figure 4.12 plots the values of the bins in each of the three colors channels. Notice how similar in distribution they are. This problem is responsible for the majority of the errors in classification for these two species. While an easy solution might be to include the size of the blobs its not quite so straightforward. The size of the blobs also depends on how close to the camera each fish is. As we want to generalize our system to other scenarios, we cannot make the assumption that the fish will always be at a certain distance from the camera.

Figure 4.10: An example of a ”yellowtailed” fish.

Figure 4.11: Detected blobs of a surgeonfish and a ”yellowtailed” (scaled up).

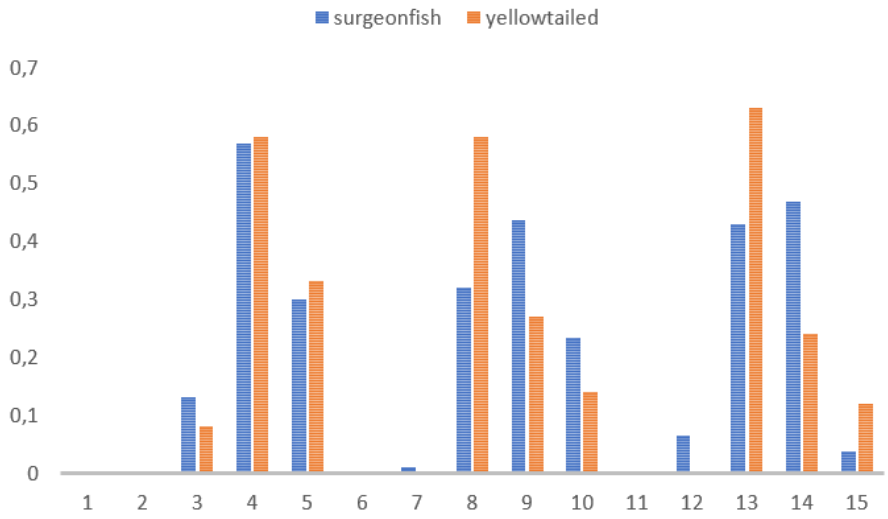


Figure 4.12: Color histogram comparison between the blobs of figure 4.11

5

Conclusion

Contents

5.1 Conclusion

In this thesis we first discussed the motivation and benefits behind 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 first tried to understand how this problem is approached in comparable scenarios, discussing the advantages and shortcomings of different methods for tracking and classifying objects in images or videos that have been previously applied.

We then proposed and implemented an approach to track and classify fish in real-time on a unique dataset built in collaboration with 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.

Tests were conducted in order to tune the parameters of our system for two different scenarios. The impact of our additions to the usual method of tracking based on detection was then quantified, with modest but positive results in the order of 2% improvements to the global accuracy. The classification methods applied also provided satisfactory results, correctly classifying 80% of the fish based on a single frame.

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.

The *Oceanário de Lisboa* showed interest in systems that would require the groundwork that we provided, for research and monitoring of their fish. We hope to have taken a first step towards helping them gain a better understanding of their animals, as we very much support their work towards conservation and educational causes.

5.2 Future Work

Some of the challenges presented in this thesis 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. Furthermore, our solution is not appropriate for fish that remain still for long periods of time, such as eels that prefer remaining partially hidden in rock crevices.

Perhaps more importantly, our testing dataset is not extensive enough to claim robust tracking and classification. Ideally we would have built a testing dataset that contains a wide variety of edge case examples through which we can analyze the degree of success of the proposed methods in a more objective manner. We were unable to build such a dataset due to time constraints.

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.

Bibliography

[1] Frank J Aherne, Neil A Thacker, and Peter I Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4):363–368, 1998.

[2] Emma Beauxis-Aussalet, Elvira Arslanova, Lynda Hardman, and Jacco Van Ossenbruggen. A case study of trust issues in scientific video collections. In *Proceedings of the 2nd ACM International Workshop on Multimedia Analysis for Ecological Data (MAED)*, pages 41–46, 2013.

[3] 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.

[4] Aaron Bobick and James Davis. Real-time recognition of activity using temporal templates. In *Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV)*, pages 39–42, 1996.

[5] Bastiaan J Boom, Phoenix X Huang, and Robert B Fisher. Approximate nearest neighbor search to support manual image annotation of large domain-specific datasets. In *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications (VIGTA)*, page 4, 2013.

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

[7] 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.

[8] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[9] Roberto Di Salvo, Daniela Giordano, and Isaak Kavasidis. A crowdsourcing approach to support video annotation. In *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications (VIGTA)*, page 8, 2013.

[10] Michael B Dillencourt, Hanan Samet, and Markku Tamminen. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39(2):253–280, 1992.

[11] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[12] 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.

[13] Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[14] Phoenix X Huang, Bastiaan J Boom, and Robert B Fisher. Hierarchical classification for live fish recognition. In *Proceedings of the British Machine Vision Conference (BMVC) student workshop*, 2012.

[15] Swantje Johnsen and Ashley Tews. Real-time object tracking and classification using a static camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Workshop on People Detection and Tracking*, pages 113–118, 2009.

[16] 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.

[17] Isaak Kavasidis, Simone Palazzo, Roberto Di Salvo, Daniela Giordano, and Concetto Spampinato. An innovative web-based collaborative platform for video annotation. *Multimedia Tools and Applications*, 70(1):413–432, 2014.

[18] 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.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[20] Alan J Lipton, Hironobu Fujiyoshi, and Raju S Patil. Moving target classification and tracking from real-time video. In *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV)*, pages 8–14, 1998.

[21] Nigel JB McFarlane and C Paddy Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193, 1995.

[22] Gayathri Nadarajan, Cheng-Lin Yang, Yun-Heh Chen-Burger, Yu-Jung Cheng, Sun-In Lin, and Fang-Pang Lin. Real-time data streaming architecture and intelligent workflow management for processing massive ecological videos. In *Proceedings of the International Conference on Social Computing (SCSM)*, pages 1074–1080, 2013.

[23] 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. 2018.

[24] Concetto Spampinato, Emmanuelle Beauxis-Aussalet, Simone Palazzo, Cigdem Beyan, Jacco van Ossenbruggen, Jiyin He, Bas Boom, and Xuan Huang. A rule-based event detection system for real-life underwater domain. *Machine vision and applications*, 25(1):99–117, 2014.

[25] 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.

[26] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2246–2252, 1999.

[27] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[28] Sébastien Villon, Marc Chaumont, Gérard Subsol, Sébastien Villéger, Thomas Claverie, and David Mouillot. Coral reef fish detection and recognition in underwater videos by supervised machine learning: comparison between deep learning and hog+svm methods. In *Proceedings of the 17th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 160–171, 2016.

[29] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[30] 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.