TÉCNICO LISBOA

Fish Behavior Detection through Video Frames and Trajectories

Gonçalo Filipe Antão Adolfo

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

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

Examination Committee

Chairperson: Prof. Name of the Chairperson
Supervisor: Prof. Alexandre José Malheiro Bernardino
Members of the Committee: Prof. Name of First Committee Member
Dr. Name of Second Committee Member
Eng. Name of Third Committee Member

October 2021

Acknowledgments

The development of this project would not be possible without the key presence of several people, for which I would like to write some appreciation words. I would like to thank my supervisors Prof. Alexandre Bernardino and Prof. Sofia Pinto for sharing knowledge and guidance throughout the thesis process, from its idea to its implementation and improvement. Thank you for sharing your experience and guiding me to the right path, but also for all the comprehension and patience. I would like to also thank Lisbon Oceanarium for the interest and trust in this project, and for letting us film videos from the oceanarium tanks that were key to the development process. A special thank you to Núria Baylina and Hugo Baptista for letting us know more about aquatic life and for helping us to understand what could be important for biologists.

Last but not least, thank you to all my family and friends for always encouraging me to achieve all the dreams that I have in mind. A very special thank to my parents Helena and José, my sister Madalena, my grandparents António and Belarmina, my godparents José and Carla, and my girlfriend and family Beatriz, Ana, Nuno, Inês, and Manuela. Thank you for all the support and comprehension throughout this project.

To each and every one of you – Thank you.

Abstract

Detecting automatically fish behaviors can be helpful to monitor fish in tanks, which can save considerable time for biologists. In this project, we developed a system capable of detecting abnormal behaviors, feeding periods, and also interesting moments to be analyzed by biologists. Due to its importance to Lisbon Oceanarium, we focused on sharks and manta rays. The system relies on video frames, and fish trajectories which are represented in a features vector format. To detect abnormal behaviors, we resorted to clustering approaches or a switching vector model, and several classifiers were trained to detect interesting moments. Additionally, it is possible to define a set of species-specific rules regarding the extracted features. Feeding periods are detected using a convolutional neural network, or based on aggregation/motion variability. To evaluate each of the approaches, several metrics are extracted such as accuracy, precision, and recall.

Keywords

Machine Learning; Behaviors; Computer Vision; Trajectories; Aquatic Species

Resumo

Detectar automaticamente comportamentos de espécies aquáticas pode ser útil para a sua monotorização, o que pode economizar um tempo considerável para os biólogos. Neste projeto, desenvolvemos um sistema capaz de detectar comportamentos anormais, períodos de alimentação e momentos interessantes para serem analisados por biólogos. Devido à sua importância para o Oceanário de Lisboa, focamo-nos nos tubarões e nas mantas. O sistema baseia-se em imagens de vídeo e nas trajetórias dos peixes, que são representadas em um formato de vetor de características. Para detectar comportamentos anormais, recorremos a abordagens de agrupamento ou um modelo de campos de movimento, e vários classificadores foram treinados para detectar momentos interessantes. Além disso, é possível definir um conjunto de regras específicas para as espécies em relação às características extraídas. Os períodos de alimentação são detectados usando uma rede neural convolucional, ou com base na variabilidade de movimento/agregação. Para avaliar cada uma das abordagens, várias métricas são extraídas, como a acurácia, precisão e sensibilidade.

Palavras Chave

Aprendizagem; Comportamentos; Visão Computacional; Trajectórias; Espécies Aquáticas

Contents

List of Figures

List of Tables

1

Introduction

Contents

1.1 Motivation

Fish behaviors themselves can be seen with different levels of abstraction. The focus can be to detect something out of the ordinary or to detect explicitly known activities such as resting, feeding, swimming, among others. The abnormal behaviors can serve as an alarm of something of interest to analyze and the known activities can help to understand fish patterns through the day (or other levels of granularity).

Feeding activity is considered especially interesting because of its impact on production costs and water quality. Underfeeding leads to aggressive behaviors while overfeeding leads to food waste (more costs) and the uneaten food/fish feces interferes with water quality. This activity is usually controlled based on observer experience, which can be subjective since many factors can contribute to fish appetite: physiological, nutritional, environmental, and management.

Detection of an abnormal behavior can mean various scenarios: fish disease, problems in water quality, poor habitat integration, or poor integration with other species. Whatever the cause, the traditional way of taking these conclusions is based on visual inspection by marine biologists. This is considered, by many, very time-consuming and highly dependent on the biologist's experience. Another possible approach, more used in the context of open sea systems, is by analyzing footage taken by divers. This method is invasive and is therefore not able to capture behaviors considered normal. One possible solution to overcome these approaches could be the installation of several cameras in the different tanks. In this case, it would be possible to continuously analyze, instead of having to be analyzed in person in the period of interest. However, the amount of data produced daily would be huge and probably there would not be enough human effort to process it.

Biologists cannot be in the various tanks at the same time. Even if there were enough team members to cover the different analysis periods, it would consume a considerable amount of time which could be used in another task. Using underwater cameras, the rate at which the footage is produced would be probably greater than the rate at which it can be analyzed by humans. Hence, there is a need to develop a system capable of helping biologists to monitor the behavior of fish.

Computer science areas, such as computer vision and machine learning, have evolved in recent years which allows the implementation of a system in a cheap way, without requiring to more expensive technology (e.g. acoustic technology).

1.2 Project Goals

The main objective of the project is to develop a system capable of helping biologists to monitor fish. To do so, it should be able to detect fish behaviors in the main tank environment of the Lisbon Oceanarium. According to our talks with biologist Hugo Batista, sharks and manta rays are species that are important to track and which are visually differentiable from the rest of the species in this tank, so they were defined

as the focus species for this project. Relatively to behaviors, the system should identify the feeding periods and if possible fishes with lack of interest, abnormal activity, and also interesting moments to be analyzed which are defined by direction and speed sudden changes, high speed, too many direction changes, manta rays swimming on the bottom, and sharks aggregation.

1.3 Contributions

In this project, we developed a system capable of detecting feeding periods, abnormal behaviors, and interesting moments through a camera and fish trajectories. For each behavior, we experimented different state-of-the-art methods, adaptions of it, and also new approaches, to move towards the best possible performance. It is possible to highlight the following points: experimentation of different clustering algorithms and the utilization of a switching vector model to detect outlier behaviors; train classifiers to be able to identify interesting moments based on trajectory features but also give the ability to define species-specific rules; train networks to detect feeding frames, but also try to rely on aggregation and motion variability. In terms of data, we defined a set of datasets along with the project development: trajectories of sharks and manta rays in the main tank, trajectories in a features vector format dataset, interesting moments ground truth and feeding frames datasets.

1.4 Outline

This document follows the traditional pipeline: related work study, system development and implementation approaches, evaluation and experimentations process, and finally achieved results. We start this document by going through several projects in the same area, or that have goals in common (chapter 2). After that, in chapter 3 we explain what was the followed approach in terms of the developed system, and also details about its implementation. Chapter 4 defines the different experimentations that were made, the tuning process regarding those, and the evaluation metrics that were calculated. In chapter 5 the results themselves are exposed and the respective conclusions about these. Finally, we make a retrospective regarding all the developed work and also present a list of possible future work (chapter 6).

2

Related Work

Contents

Modelling trajectories allows the identification of activities and the analysis of behaviors. Over the years several projects have been developed within the scope of trajectory analysis, and in different contexts: analysis of tourist activities as the example given in [1] by using semantic information; modeling of pedestrian/car/cyclist trajectories based on Markov models [2]; methodologies without a concrete domain [3] (clustering trajectories based on shape); among others.

In the context of this project, the domain of interest is aquatic species. The related work presented in this section is divided as follows:

- modules needed for the acquisition of trajectories: fish detection, tracking for several consecutive images and, in certain scenarios, classification of the image of the detected fish;
- detection of abnormal behaviors: many of the applications developed are intended to help biologists, detecting something out of norm;
- recognition of certain activities to assist monitoring (feeding, swimming, resting, among others).

2.1 Detection Tracking and Classification

2.1.1 Previous Projects

Our project is being developed following previous projects [4, 5]. Castelo et al. explored the problem of detection, tracking, and classification of aquatic species within the scope of the Lisbon Oceanarium. Detection used background subtraction methodologies. In this approach, a background image is estimated by averaging the value of sample images. Subtracting a certain image from the calculated image allows the identification of motion pixels, and using a given threshold to obtain a binary image, where the active pixels are the motion pixels. Through this approach, it is possible to label the regions that are connected, which are considered the regions of the fish detected at that moment. Using the set of blobs identified in two consecutive images, the association is made through the characteristics of these regions, both the position of the centroid and the predominant colors.

The detection module is not perfect, so associations can be lost. To try to deal with this problem, an algorithm (Kalman Filter) is used to predict the next positions of a given fish and a heat map is calculated around the estimated position. If there is a region with a similar area and similar colors, it is considered the region of the fish in question, although it has not been detected as a movement zone. In this project, the method used to classify depends on the environment. In the case of the coral tank, the class associated with a given fish is given by the one with the most overlap between color histograms. On the other scenario, in the main tank, since the fish have very similar colors to the background color, the histogram of oriented gradients (HOG) was chosen: $\theta(x, y) = \arctan(\frac{G_y(x,y)}{G_x(x,y)})$ where G_x and G_y

are the image gradients for each dimension. Several linear regressions are used to reconstruct the histogram and the class associated with the model weights that best reconstruct is the one assigned.

Santos et al. [5] aimed at improving tracking. The idea was mainly adding robustness to tracking problems resorting to: color history, temporary tracks and motion prediction. When two distinct fish regions overlap, their histogram get corrupted. To overcome this problem, the color histogram takes into account the color histogram from several previous frames and not only from the last one. The temporary track concept defines that a new track is marked initially as temporary, and when it remains for 5 frames then it turns into permanent. This concept allows to not propagate noise tracks, related for example from the division of some fish region into two different regions. Finally, a Kalman Filter is also used to predict future positions of a given fish, and the predicted position is the one used for the position similarity calculation. This can be useful in scenarios where the true track starts following another fish or when there are miss detections.

2.1.2 Detection

In the [6–9] approaches, background subtraction methods are used. However, an application in [6] allows the user to choose the highest contrast color plane, according to the Red-Green-Blue (RGB) color plane, instead of using the gray-scale image. Additionally, the background image is only updated every 100 seconds. The methods in [8,9] use the Multi-Scale Retinex contrast enhancement algorithm before background subtraction is applied. This algorithm enhances the contrast between regions of the image: the original image passes through a Single Scale Retinex: $SSR(x, y) = \log(I(x, y)) - \log(G(x, y, \alpha) \circledast I(x, y))$ where I is the original image and G a 2D Gaussian filter with α scale, $\circledast$ represents the convolution operation. Multi-Scale Retinex is defined as the application of several SSR's with different scales in the Gaussian filter. The work described in [7] also defines a methodology, an alternative to background subtraction methods, based on Optical Flow, since the focus of this paper is not the detection of fish regions, but the analysis of the school itself. A particle grid is placed over the image and between consecutive images, its movement is estimated. The detected movements are assumed to be the movements of the fish. Although this method is not subject to problems such as changes in luminosity, which have an impact on the background and consequently on the obtained motion regions, it presents greater temporal complexity for motion estimation and is also not resilient to problems such as small movements of background objects.

In [10] and [11], two different approaches are used to detect fish present in a given image. The first uses a Gaussian Mixture Model while the second uses a YOLO network. The idea behind the first method is to model the intensity of each pixel by a set of Gaussians, and the training works as follows: for a new pixel value, the parameters (average, standard deviation, and associated weight) of the Gaussian that best represent its intensity are updated. Given a new image and the set of Gaussians for each pixel,

all pixels in the new image are classified as being part of the background or not. The advantage of this approach is the ability to model small movements of background objects.

The other method (YOLO) is part of the recent field of convolutional neural networks. Despite being applied in several domains, they are specially useful to model structure. In an R-CNN network, it is used a process called Selective Search: several bounding boxes (of different sizes) are placed over the image domain and are grouped according to similar characteristics, of adjacent bounding boxes, generating a set of proposed regions that are passed as input to another convolutional neural network (CNN). The YOLO algorithm achieves better training and classification times: it starts by dividing the image into a grid of SxS regions and m bounding boxes are generated in each region; for each bounding box, the network returns as output the probability of belonging to each class and only those with a value greater than x are considered to locate the object. In this sense, with the YOLO algorithm, a single CNN is used to predict bounding boxes and calculate their respective probabilities. The use of this algorithm is more resilient to common problems such as changes in the background and occlusion of fish, but it does not return the fish mask, only its bounding box. However, in general, it presents better detection results since it is more robust.

2.1.3 Tracking

Regarding tracking on consecutive images, [11] uses an approach based only on distance while [10] uses the Adaptive Mean Shift algorithm. In the first approach, given a new $f+1$ image, it checks if there is any close detection in the previous image f. If so, the same identification is associated, otherwise, a new identification is associated. With the AMS approach, the idea is to move the region of interest (in this case of a certain fish) in the direction of the highest density area until it converges. For this, the binary image of the current instant and the region of interest detected in the previous image is used. With this, the centroid of the zone is calculated given the binary image, the zone is moved towards the centroid, and this is repeated until the successively obtained centroids converge. Finally, it is verified in which blob the final centroid is inserted and the same identification is given. Both approaches do not correct occlusion problems. However, using an image such as the histogram back-projection image makes the second method more robust to this type of problem.

The project described in [12] proposes the innovative idea of identifying and tracking sharks based only on the characteristics of their dorsal fine as if it were a fingerprint. Two convolutional networks (CNN's) are used: one that detects sharks present in a given image and the second that detects the region of the shark's dorsal fine, given its bounding box. In order to not take into account the background pixels of the fine region, a K-Means algorithm is applied. The edge of the dorsal fine that presents greater disturbances is the edge that best identifies the shark. To obtain it, first, it extracts the triangle that best characterizes the fine contour and the midpoint (of each edge) that is farthest from the real contour is the

point belonging to the edge of interest. Then, k key points are identified on this edge of the fine contour and they will be defined as the boundary descriptor. These are the points where there are the biggest disturbances (salient locations) given by: $p(u_i) = D(u_i, m, \sigma) - D(u_{i+1}, m, \sigma)$ where u is a contour point and D a corner response function being m the multiplication factor and σ the standard deviation of a Gaussian filter. Since each shark is defined by a vector of K points, these are first normalized to be invariant in scale and orientation and passed to an algorithm based on graphs, given two sets: one from the sharks detected in the current frame and another from the previous frame (or even other images). A connected full graph is built, and the weights between nodes are given by the Euclidean distance. Finally, a minimum-weight perfect matching algorithm is applied to obtain the matches. Under this approach, it is important to highlight the need to use high resolution images, which may be difficult to achieve.

2.1.4 Classification

Research described in [10, 13, 14] also address classification. In [10], each fish is represented by a vector of texture and shape features. Regarding texture, statistics of the gray histogram and the co-occurrence matrix are extracted: contrast, energy, correlation, among others. The shape is defined by the histogram (with 30 cells) of the discrete Fourier transform (DFT) on the contour points and the first 20 local minimum of the Curvature Scale Space (CSS) image. The set of example feature vectors is passed to a Principal Component Analysis (PCA) algorithm, which allows reducing the dimensionality by projecting the points on the axes of greater variance. The vectors, of reduced dimensionality, are finally passed as input to a linear discriminant.

The project [13] uses the size of the fish together with the size of the different fines: anal, caudal, dorsal, pelvic, and pectoral. A Support Vector Machine (SVM) is used, because it is more robust than a linear discriminant since it calculates the hyperplane that best separates the different classes, with the optimal margin. The real estimation of the size of the fines can be a challenge, but it can be done through semantic segmentation (Mask Neural Networks), as described in [15]. Finally, [14] uses deep learning for classification between different species. A convolutional neural network (CNN) is trained using a very large dataset of sample images for several species. This type of methodology has the disadvantage of needing a lot of examples, given the high number of parameters. The described method has the particularity of receiving as input the binary image of the fish concatenated with the 3 Red-Green-Blue (RGB) planes. This is calculated by applying the OSTU algorithm [16] to the gray image, to discover the threshold that better divides the intensities present in the fish image (less intraclass variance and greater interclass variance). To enhance the fish region, morphological operations are also applied (opening and closing). The results obtained with deep learning were better when compared to previous approaches, according to the accuracy obtained in test sets.

2.2 Abnormal Behavior Detection

2.2.1 Non-Machine Learning Methods

Beyan et al. [17] proposed a method to filter a set of trajectories considered normal from a total set of trajectories. This mechanism is based on the application of several rules that characterize normality. They are applied in the form of a cascade: the total set of trajectories is passed as an entry to the first rule, which in turn filters out a certain part (according to that rule) and the rest are passed as an entry to the next. Trajectories that are not filtered by any of the rules are considered abnormal.

The fish, when caught by the camera's ray of vision, do not have many changes in direction. This is the basic idea of the proposed rules. It is argued that usually there are no more than three changes between primitive movements: horizontal movement, vertical movement, and stationary movement. Each of the rules models the possible set of moves giving a total of $3 + (3 \times 2) + (3 \times 2 \times 2) = 21$ rules. Each filter, and for each of the primitive movements, is also associated with the concept of "search area" (calculated by analyzing example trajectories). The trajectory must satisfy the following conditions:

- all the centroids of the bounding boxes of the fish, during a certain movement, must be within a certain area ("search area");
- the centroid in a given frame f must be within an area given the position detected in frame $f - 1$.

The results illustrate that this approach has a low false-positive rate. However, most of the trajectories did not pass any rule, giving a high rate of false negatives. That said, the approach has not shown good results when it comes to detecting abnormal trajectories but it can work as a preliminary method for another approach.

The method defined in [7] aim at detecting special events in schools, instead of focusing on certain fish individually. The underlying idea is that the fish in a school does not disperse much in terms of movement. The proposed kinetic energy model, according to physics, represents the energy that a given object presents given its movements. It combines two measures of the dispersion regarding the motion vectors, the velocity and the direction angle: $E_{kn} = (D + 100)^2 \times (-E)$ where D is the dispersion regarding the motion vectors and E the dispersion concerning velocity in relation to turning angle. Given several example images, a threshold is estimated for each measure: dispersion of movement, dispersion of velocity/angle, and energy value. When these conditions are met for n consecutive images, an alarm is given.

The features of the school are modeled by two histograms: a 2D histogram of the motion vectors (in x and y), and a 2D histogram of the velocity concerning the direction angle. These are normalized and the entropy value is extracted: $E = -\sum_i^{m_1} \sum_j^{m_2} p_{xy}(i,j) \log_2(p_{xy}(i,j))$ where p_{xy} is the probability in a given cell, m_1 and m_2 are the number of bins for each feature. This measure is associated as a

measure of uncertainty/chaos. The higher the entropy value, the more dispersed the movement features of the school fish are, indicating something special. This approach, as it uses background subtraction methodologies to detect fish, is very sensitive to occlusions and unions of different fish regions. For this reason, it also proposes a method based on Optical Flow, as explained in the previous section.

2.2.2 Machine Learning Methods

The task of detecting trajectories related to something abnormal can also be seen as a problem of outlier detection or even a classification problem. In [10], a clustering algorithm (IKMeans) is applied over the entire set of trajectories and the clusters with few samples are considered to be trajectories of interest (when compared to the total number). This algorithm is applied to each set of trajectories of each species since different species have different motion patterns.

Trajectories can contain a large number of points, not all of which are needed to describe it correctly, which can lead to a long processing time. To try to solve this problem, before being used as input to the clustering algorithm, they are pre-processed with the Douglass-Peucker algorithm. On this step, the goal is to remove points from the trajectory that have no impact on its original form. This is done by recursively dividing the trajectory. It consists of the following steps:

1. mark the first and the last point as ”to keep”;
2. obtain the furthest point from the line segment AB, with A initially being the starting point of the trajectory and B being the last point;
3. if the distance is greater than ϵ then that point is marked as ”to keep” and the algorithm returns to step 2 but with the line segments A-FurthestPoint and FurthestPoint-B. Otherwise, all points in the middle are discarded.

The IKMeans clustering algorithm itself also tries to deal with this issue. It takes advantage of the Haar Wavelet decomposition, which allows representing a time series of the trajectory in lower resolutions, preserving the original information/form. The idea of the IKMeans algorithm is to apply the traditional KMeans algorithm in the set of time series of the trajectories at increasing levels of detail. This approach usually converges soon at low levels of resolution. However, even if that does not happen, the centroids of one level will be the centroids of the next level, allowing them to converge more quickly in the next level. Overall, better results are achieved in terms of clusters' quality and processing time than the traditional algorithm.

The method described in [18] requires a clustering algorithm to train a hierarchical classifier. The points at a given level of the tree are obtained as follows:

1. feature extraction (subset) and Principal Component Analysis (PCA) application;

2. clustering and outlier detection;

3. feature selection: increase features subset;

4. return to step 1.

Despite being a model whose training is based on clustering, the example trajectories must have an associated class. The previous points are repeated until eventually, the accuracy goes down with the increase of features space. When this happens, the clusters obtained that are pure (only normal trajectories or only abnormal trajectories) are maintained at this level. The trajectories belonging to the remaining clusters are used to estimate the clusters of the next level of the tree, thus the steps mentioned are repeated but with different input. In this way, each level of the tree will have a set of associated pure clusters and considers different subsets of features. Given this tree the classification is made as follows:

1. for each level, the closest cluster to the new trajectory is obtained (given its feature vector for the level in question);

2. it is verified to which class each of the closest clusters are associated;

3. the new trajectory is classified as normal if there is a pure cluster of normal trajectories as the closest cluster (in at least 1 of the levels).

The [11] approach also trains and evaluates various classifiers to classify trajectories as normal or abnormal. However, what is passed as input in this method is an image with all the trajectories of the fish detected in a 10s window. Several models were evaluated, namely: Naive Bayes, K-Nearest Neighbors, Linear Regression, and a Random Forest. Within the models described, the Naive Bayes probability model was the one that achieved better results. Since the input of these models is the image itself, the usage of deep learning was expected. However, the decision must have been derived from the reduced set of example data. Both approaches obtained similar values in terms of evaluation despite the [18] method being tested on more robust datasets, including datasets from other domains. Additionally, it takes into account that the trajectories may not be perfect: an interpolation is applied to derive certain missing points. The other approaches do not take this problem into account, as well as the fact that abnormal events can be quite fast. In this case, in longer trajectories, these events can be canceled by the normality of the remaining segments.

2.3 Activity Recognition

Papadakis et al. [6] developed a system capable of monitoring a set of fish tanks simultaneously. The project aimed to observe fish behaviors in the following environment: a net was placed in each tank to

separate its area into two zones, and all fish were placed in only one of the regions. The tanks had different fish densities and the net in different states. The focus behaviors were inspection and biting.

The inspection activity is detected when at least one fish approaches sufficiently the net and it is counted by the number of images in which it occurs. The midpoint of the network is manually marked as a reference point by the user. For each image, the horizontal distance between the center of mass of each fish and the reference point is calculated. Other statistical metrics are also extracted (to analyze certain correlations with the activities): center of mass of the school, an average of the horizontal/vertical distance, and average velocity. With this approach, it was possible to take conclusions, about inspection and biting, in different environmental conditions. Despite being defined in the article, it is not explicit how to detect the biting activity.

The approaches in [19,20] used an accelerometer on each fish of interest, and certain activities were detected based on the collected values. Broell et al [19] (2013) focused on detecting the following set of activities: swimming, feeding, and escaping. They proposed a signal processing system based on the analysis of time series features, related to the acceleration in each dimension. Multiple features were calculated, based on example data for the different behaviors, in several domains: frequency (spectral and wavelet analysis), probability (mean, maximum, variance), and time (autocorrelation, integrals, and derivatives). The idea was to identify which features could have identical values within the same activity but different between different activities. They managed to conclude the following:

- magnitude of acceleration $MA = \sqrt{x^2 + y^2 + z^2}$ standard deviation was the most efficient feature to distinguish the swimming activity from the others, presenting significantly lower values during this activity;

- the distinction between feeding and escaping activities achieved better results using 6 features: 4 of them are associated with the comparison of values between x and y dimensions (standard deviation, maximum value, acceleration range, and root mean square) and the remaining 2 are correlation values (autocorrelation of the acceleration norm, and spearman correlation between x and y dimensions).

In short, this method has a total of 7 parameters: a threshold for each of the mentioned features. These values were estimated in a different set from those used for analysis and in a brute force way: go through a set of values and check which ones presented the best division between activities. The precision of the division is associated with the weight of that feature for the final decision, which can be interpreted as a confidence value in the parameter in question. With the first parameter, the signal processing method first identifies ”fast start” events (feeding and escaping). It is used a sliding window covering each 1 second period of the acceleration norm, and the standard deviation is calculated. When the condition is verified, then the remaining parameters are used to distinguish between feeding and

escaping. Finally, in this project, the impact of frequency was also studied, that allowed to conclude that lower frequencies give worse results. However, in certain scenarios, it can be advantageous in terms of energy costs. It is a context-dependent tradeoff.

Zhang et al. [20](2019) solves a similar problem but focuses on activities related to sharks: swimming, resting, feeding, and non-deterministic movement. Similarly to the previous method, it uses time series of the value of the overall dynamic body acceleration (ODBA), calculated by adding the absolute value of the acceleration in each of the dimensions. Given a set of example time series for each activity (2-second segments), three different models of deep learning are trained, more specifically convolutional neural networks:

- Shark VGG1: two convolution layers followed by a max-pooling, repeated 2x and always using twice the filters of the previous convolutions. After feature extraction layers, the output is full connected to a layer with 1000 neurons which in turn is full connected to a layer with 100 neurons. Finally, the last hidden layer is connected to an output layer, with a softmax activation function, and four neurons, which is the number of activities that are intended to be detected;
- SharkVGG2: the previous model uses 1x5 kernels. In this version, the same layers are used for feature extraction but with different kernel sizes (1x3, 1x5, 1x7). The output, the this three kernels, is concatenated and passed as input to the first hidden layer. In other words, the layers related to feature extraction are applied with different kernels “in parallel”.
- SharkInception: uses the concept of inception. In this case, the feature extraction layers are not replicated “in parallel” but in each convolution block, before being passed to the max-pooling layer.

The application of several kernels allows processing at different levels of detail in parallel. The results shown in the article demonstrate that all networks went into overfitting since the accuracy obtained in the train set was much higher than the one obtained in the test set. It should also be noted that approaches based on this type of sensors are considered invasive by certain specialists and that are easily lost from the animal, causing additional economic costs.

Zhou et al. developed several projects [8,9,21] within the scope of the feeding activity. Initially, in [8], the goal was to detect the feeding activity through the analysis of the level of aggregation of the school, since it is usually higher during this period. This measure is described using a value called Flocking Index of Fish Feeding Behavior (FIFFB). For its calculation, fish are first detected in a given image, and their center of mass is calculated. Given the set of centroids, the Delaunay Triangulation algorithm is applied to originate a set of triangles that interconnect the points. This algorithm allows obtaining the triangles that maximize the minimum angle of all triangles. It is defined by the following points:

1. a point p_1 is selected randomly from the set of points not covered yet;

2. the first edge is formed with the closest point p_2;

3. the third chosen point is the point that forms the circumference with the smallest radius and that satisfies the Delaunay rule: no point can be within circumferences formed by other triangles.

With the obtained triangles, the FIFFB value is described by adding the perimeter of all triangles ((2.1) where n is the total number of triangles and L the length of each side). The lower this value, the greater the level of aggregation. The analysis of the value of this index also makes it possible to indirectly conclude the level of appetite: the longer the fish take to return to their "normal" aggregation level, the longer the feeding period and in turn the greater the appetite.

$$FIFFB = \frac{\sum_i^n (L1_i + L2_i + L3_i)}{n} \tag{2.1}$$

The idea of this approach possibly works best in scenarios where the species in question are not usually aggregated, so the difference is greater during the feeding period. In [9], one more index is used: Snatch Intensity of Fish Feeding Behavior (SIFFB). In this method, it is argued that fish usually eat close to the surface, and during the feeding period, the surface texture changes substantially due to the intensive movements of the fish. The texture is represented by the gray level co-occurrence matrix (GLCM). This matrix is a $p \times p$ matrix where p is the number of possible intensity values. For each position (i, j), the corresponding value defines how many times these values occur together given an offset in terms of x and y. The SIFFB index value is defined as the contrast value which is higher during the feeding period. It is given by: $\sum_i^p \sum_j^p (i-j)^2 P(i,j)$ where i and j are intensity values and P the GLCM. In this article, it is also proposed a methodology, based on a model that combines a fuzzy inference system and a neuronal network, to identify when the feeding should be stopped. The input of this model is the two described indexes and the output is a binary value: to stop or not.

Finally, in article [21], an innovative idea is described to identify the appetite of the fish present in a given image. A convolutional neural network (CNN) is trained based on several images at different levels of appetite. The dataset was increased using transformations of rotation, scale, and translation once large amount of data is needed to achieve good performance. The idea in this approach is that the model itself learns image patterns associated with each level of hunger. The network architecture consists of two series of convolution layers (using 5x5 kernels) and max-pooling (using 2x2 kernels), followed by two hidden layers (with 120 and 84 neurons respectively), and an output layer with 4 neurons representative of the various hunger intensities. This approach may have worse performance when several species are detected by the camera, in environments considered more complex. Additionally, the measurement of appetite itself can be considered subjective, from specialist to specialist.

2.4 Summary and Conclusions

2.4.1 Detection

Background subtraction methodologies can be simpler to implement. However, they are very sensitive to environmental changes. Gaussian mixture models can deal with small background movements once it can model different pixel intensities. Finally, Convolutional Neural Networks are the most robust method to this problem but they need a huge amount of data to achieve good performance.

2.4.2 Tracking

The base approach for tracking is position-based. This can be very sensitive to occlusions derived from fish crossing movements. More features can be used to make this module more robust namely: color, shape, and texture features. Motion prediction can also be useful to recover from missing detection or occlusions during a few frames. In situations where the actual prediction and full track are not important, optical flow can be a good choice to have an idea of the general movement (for example using a particle grid).

2.4.3 Classification

To classify the fish frame, there are two options: feature extraction plus a model/set of rules or an approach based on neural networks. Convolution Neural Networks usually achieve better performance but, as said previously, they need a considerable amount of data. The traditional feature extraction approach can also achieve good results, resorting to color, shape, and texture features.

2.4.4 Behavior Detection

Table 2.1 defines a summary for the behavior related projects. From that, the following limitations are concluded:

- the rule-based approach [17] gave a substantial amount of false abnormal behaviors due to the rules coverage and the different species taken into account;
- [7] method has the disadvantage of focusing on shoal behavior instead of fish individually;
- clustering approaches [10, 18] are applied considering all data points of the trajectory, so the interval related to the abnormal behavior can not be gathered. Additionally, the normality of the most part of the trajectory can cancel some fast-start abnormal event;

- projects that used sensors [19, 20] are considered invasive and these are easily lost causing more costs.

Approach	**Behaviors**	**Limitations**
Motion rule-based [17]	Normal/Abnormal	Lots of false abnormal behaviors
Kinetic energy (velocity + turning angle) [7]	Normal/Abnormal	Shoal based
Clustering and outlier detection [10, 18]	Normal/Abnormal	Performed on all trajectory
Reference Zones [6]	Inspecting/Biting	Image plane (2D)
Accelerometer data [19, 20]	Swimming/Resting/ Feeding/Abnormal	Sensor costs, invasive, easily lost
Flocking and Snatching Index [8, 9]	Feeding	Species must have a flocking behavior during the feeding period

Table 2.1: Behavior Detection Summary

3

Implementation

Contents

3.1 Methodology

The proposed solution followed the previously developed projects in the Lisbon Oceanarium domain. Castelo et al. [4] explored the detection, tracking, and classification blocks, while Santos et al. [5] experimented several detection approaches and improved the tracking algorithm. With these projects, we can detect fishes in a given image, and also follow them through consecutive frames. Given a set of trajectories, which are the output of the tracking module, we can now try to understand something related to fish behavior.

As defined previously, the goal of this project is to be able to detect a set of behaviors: anomalies, feeding periods, and interesting moments. Anomalies and interesting moments rely on trajectories data because their approaches depend on the features that are extracted from those. However, to detect feeding periods we only rely on the images of the video, it is not necessary to know the fishes that are present on each frame. Figure 3.1 illustrates the bridge between the previous works and the current goal of detecting fish behaviors. In this figure, we can verify the described inputs and outputs of each of the blocks.

Similar to the previous projects, this system requires that the camera stays at a static position, mainly because of the trajectory definition which is characterized by a sequence of 2D positions.

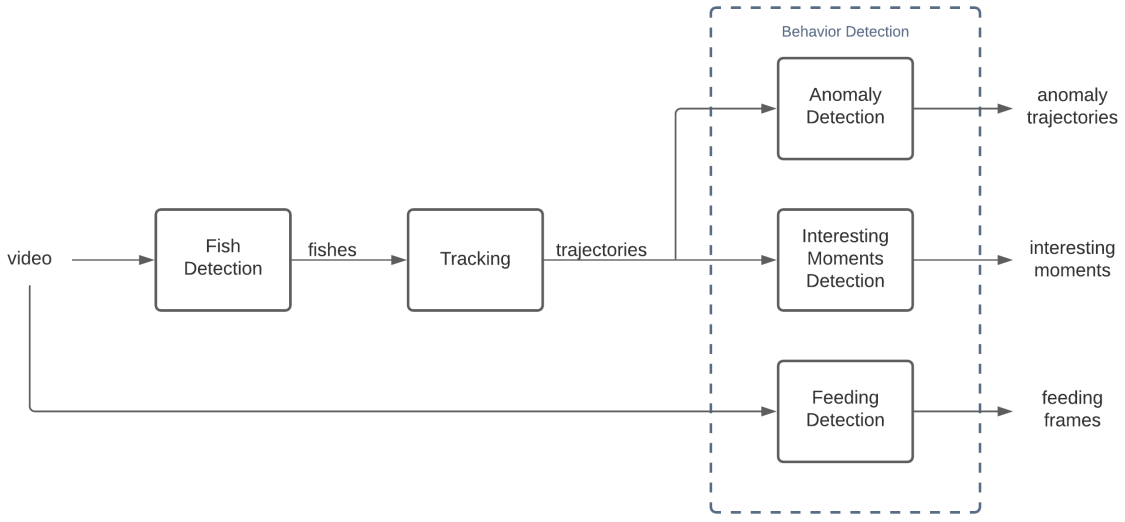

Figure 3.1: Conceptual flow of behavior detection

3.2 Library Modules and Integration Architecture

The Behavior Detection Module (BDM) is a library that was developed with the goal of providing a set of approaches to handle the fish behavior detection task. As explained before, we are interested mainly in some pre-defined behaviors: anomalies and interesting moments, feeding period and trips to the surface because of the swallow air activity. Figure 3.2 illustrates the several packages that constitute this module. Through this image, we can already verify the different approaches and features that can be used for each behavior. Relatively to the development organization, the different behaviors were divided into different packages. The remaining packages implement complementary functionalities that are used for the behavior detection task:

- trajectory reading: be able to read trajectories from a file, and parse to a suitable format;
- pre-processing: be able to fill possible gaps that may exist on the trajectory, to smooth the trajectory path, and to segment it on different partitions;
- trajectory features: be able to extract features from the trajectory, such as speed, turning angle, etc...

These capabilities are not directly intertwined with the behavior itself but were used preliminary to facilitate its detection. Interpolation can be especially important when the algorithm is impacted by missing detection. Additionally, trajectories can also have some noise due to the bounding box noise, so smooth its path can be important to have more reliable feature values. Segmentation did not have such an important rule as the blocks described before but was used to verify a hypothesis: classification of the different partitions instead of the trajectory as a whole. Finally, almost all the methods depend on the features that are extracted from the trajectory, so this package was extremely important to the main packages, as we can state from the package diagram. From this diagram, we can also visualize the different approaches to detect each behavior:

- anomaly detection: clustering-based, and switching vector model;
- interesting episodes: trajectory classification, rule-based, and also switching vector model;
- feeding: aggregation-based, motion-based (active pixels identification, and optical flow), and convolutional neural network (CNN);
- swallowing air: position-based.

Anomalies can be defined as something that escapes from the normal state. To detect something of this nature, we can group all the samples and verify which of the groups represent outliers (clustering-based) as in [10], or model the set of trajectories considered as normal and be able to verify how well a

new trajectory can be described by this model (switching vector model [2]). On the interesting episode's side, these can be described by a given set of specific criteria, as opposite to outliers. We can either classify each trajectory or trajectory segment as being interesting or not based on machine learning models, declare a set of rules based on its features as in [17], or use an adaption of the switching vector model. For the feeding behavior, we also have several approaches: based on the aggregation of the several detected fishes as previously done in works [8, 9], based on the motion variance, or using a convolutional neural network similar to what was done in projects [19, 20].

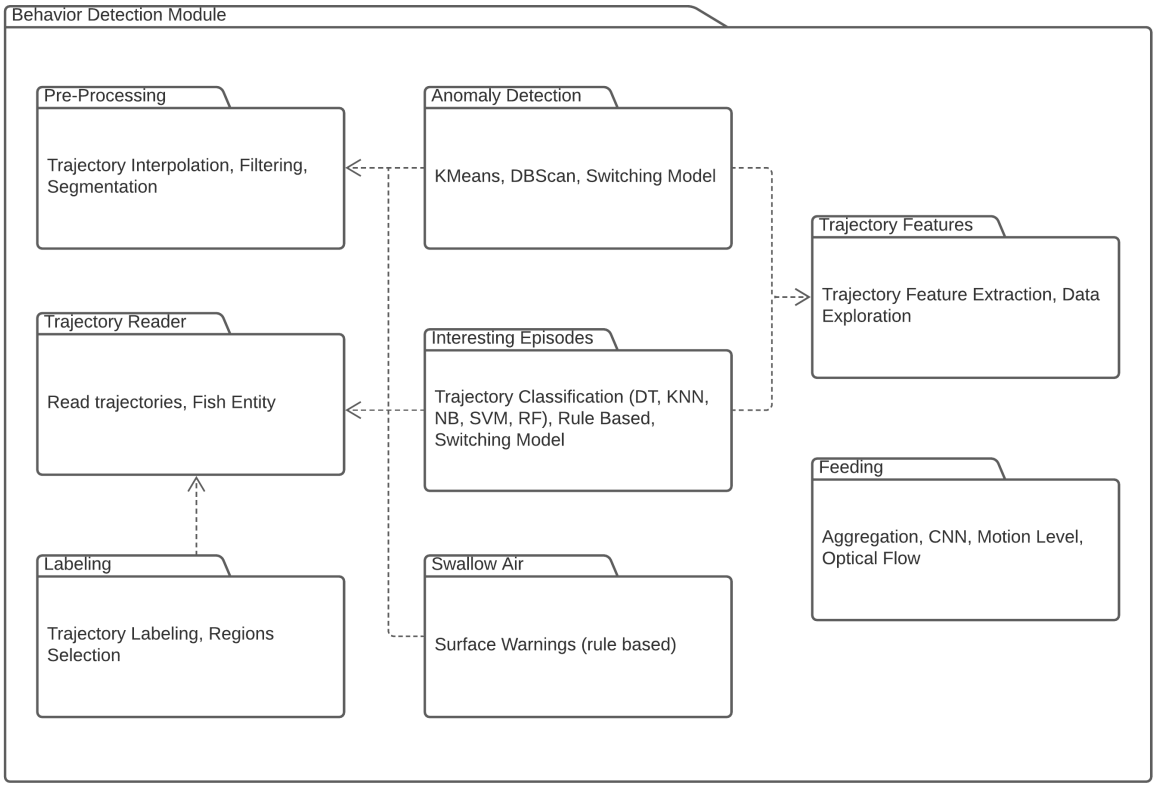

Figure 3.2: Behavior Detection Module (BDM) packages

As described previously and as we can state from the package diagram, the main goal of this library is to be able to detect fish behaviors, so this can be integrated into a given system. Figure 3.3 illustrates a possible architecture where this software development kit (SDK) could be incorporated. We can see it highlighted in red on the worker nodes. The general flow of this system can be described by the following points:

1. frames are captured from a given tank of fishes;
2. from time to time, a server will receive a portion of frames. It will be stored on the file system and sent to a worker node;

3. this worker node will apply all the logic regarding behaviors and retrieve all the information relative to those, and these will be sent back to the server;

4. the results are stored on some kind of structure.

Finally, and most importantly, we have the end-users (biologists) at the edges of the system. These will have the ability to analyze all the results and take all the necessary conclusions through the use of an application (it could be a mobile, web, or even desktop application).

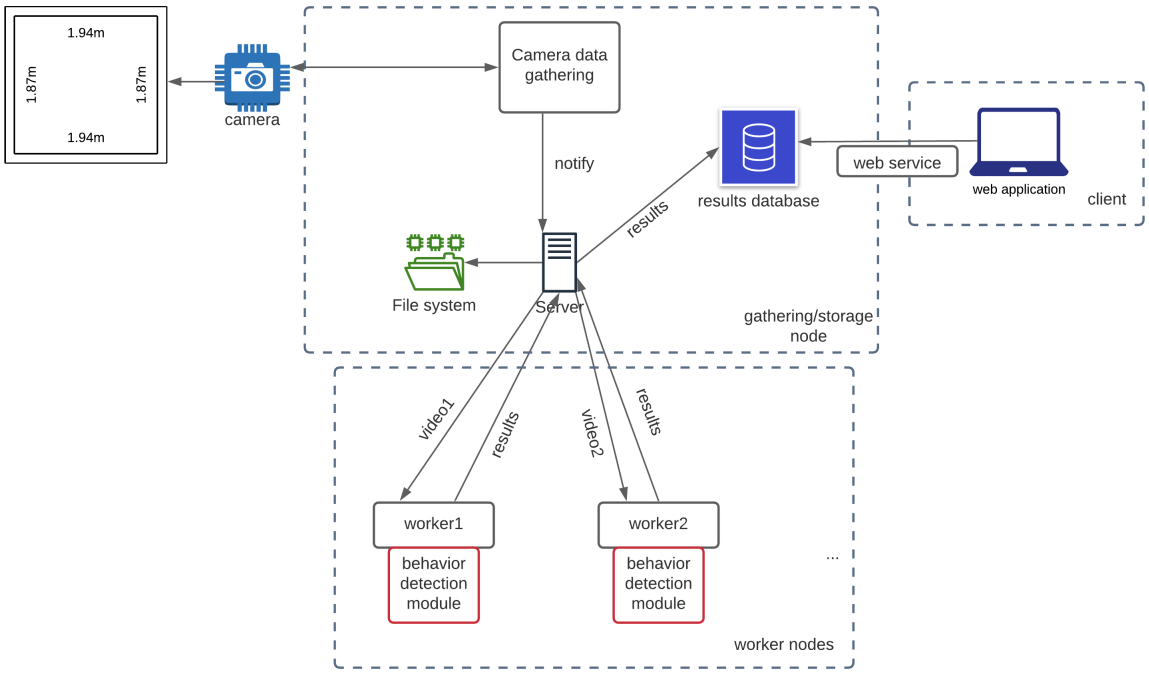

Figure 3.3: Example of a possible integration architecture

3.3 Trajectories Processing

3.3.1 Pre-processing

Trajectories are not perfect. Detection and tracking algorithms can also have errors. It is possible to miss the detection of some fish in a given instant, and it is also possible to fail its tracking especially if the fishes are very aggregated. In the pre-processing phase we try to overcome some of these problems by applying interpolation to fill possible gaps in the trajectory, and also exponential filtering to reduce the noise of the path due to the different fish angles and bounding box identification. Additionally, it is also possible to segment the trajectory into different partitions.

Interpolation

An interpolation was the applied solution to fill the gaps that a trajectory may have. Two types of interpolation were implemented: linear as used in work [18], and also newton. This is applied for both axes independently: x position and y position.

Linear interpolation assumes that the speed during the missing period is constant. The value for one axis (x or y) for a given instant t can be calculated as

$$y(t) = y_0 + (t - t_0)\frac{y_1 - y_0}{t_1 - t_0},$$

where t_0, t_1 are the timestamps of the gap edges positions, and y_0, y_1 are the position values (x or y) for those points. On the other hand, newton interpolation takes more points into account (interpolation points). Using $k + 1$ interpolation points, defining a polynomial of degree k, the value for an instant t can be calculated as

$$N(t) = \sum_{j=0}^{k} a_j n_j(t),$$

where a_j is the jth coefficient and n_j is the newton basis function. In the newton interpolation approach, there are two different ways of choosing the interpolation points:

- equidistant points among the position time series;
- nearest points to the gap edges.

Figure 3.4 illustrates an example of the application of the different methods of interpolation. The generated synthetic gap is surrounded by red dots. We can verify how each of the approaches predicts the missing position points. As expected, the newton interpolation takes more into account than the two edge points, since the coefficients are calculated with Newton's divided differences, so the predicted line is not straight. The downside of this polynomial is the Runge's phenomenon which can be slightly attenuated using the nearest points of the gap as interpolation points.

To have an idea of the best interpolation approach on our domain, synthetic gaps were generated and predicted using each of the methods, and the mean square error was calculated. Figure 3.5 illustrates the results for this experiment. Overall, a simple linear interpolation gave better results which can be explained by the linear motion of the focus species. Newton interpolation using the nearest points as interpolation points also obtained similar values, although these points can reflect a wrong trend sometimes that can lead to a higher error.

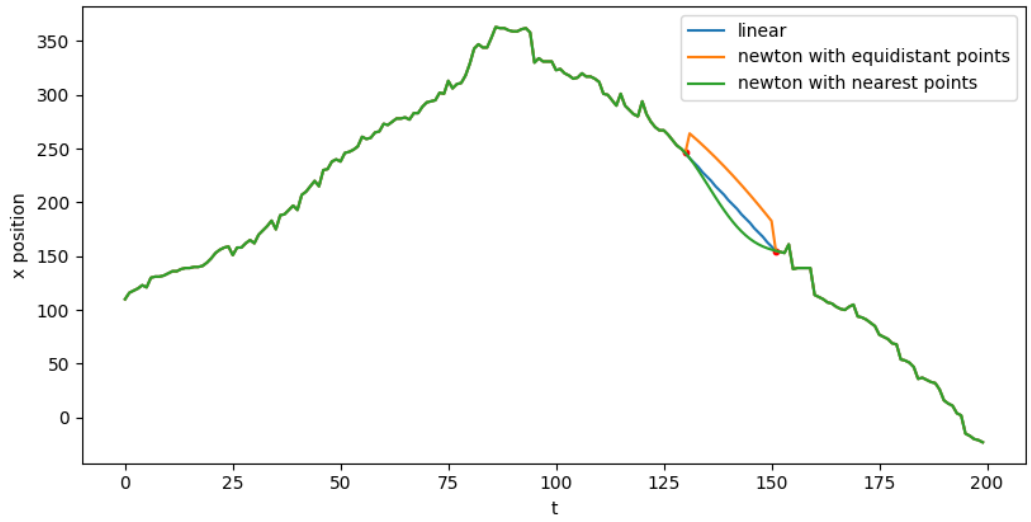

Figure 3.4: Interpolation methods comparison

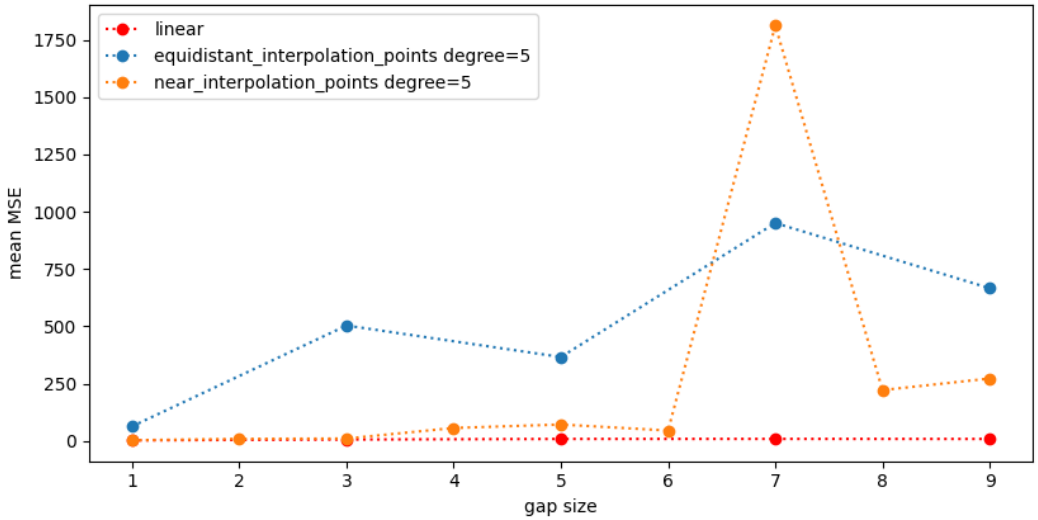

Figure 3.5: Interpolation performance

Filtering

The trajectory points can have noise due to instability on the detection task. Most of the time, the points are characterized by the center of the bounding box detection and this can be noisy because of the image plane and fish aspect ratio changes. One of the possible solutions is the application of moving average techniques to smooth the position time series. The exponential moving average was the chosen filter, so closer data points can have a higher impact. The new value for a given instant t can be calculated as

$$y(t) = \sum_{i}^{N} w_i y_{t+i},$$

where N is the window size, and w_i is the weight of the point at distance i. This filter also depends on the definition of a α value which determines how the weights are attenuated among the window.

Figure 3.6 illustrates the difference for each axis before and after the application of the exponential filter. Additionally, Figure 3.7 also complement this problem in a trajectory path visualization. The left frame in the figure draws the trajectory using the original positions. On the other side, the right frame draws the trajectory after applying the exponential moving average filter through the x-axis and y-axis positions. The trajectory after the filtering process is significantly more smooth, which will reflect also on the feature extraction phase, and consequently in the following behavior detection processes.

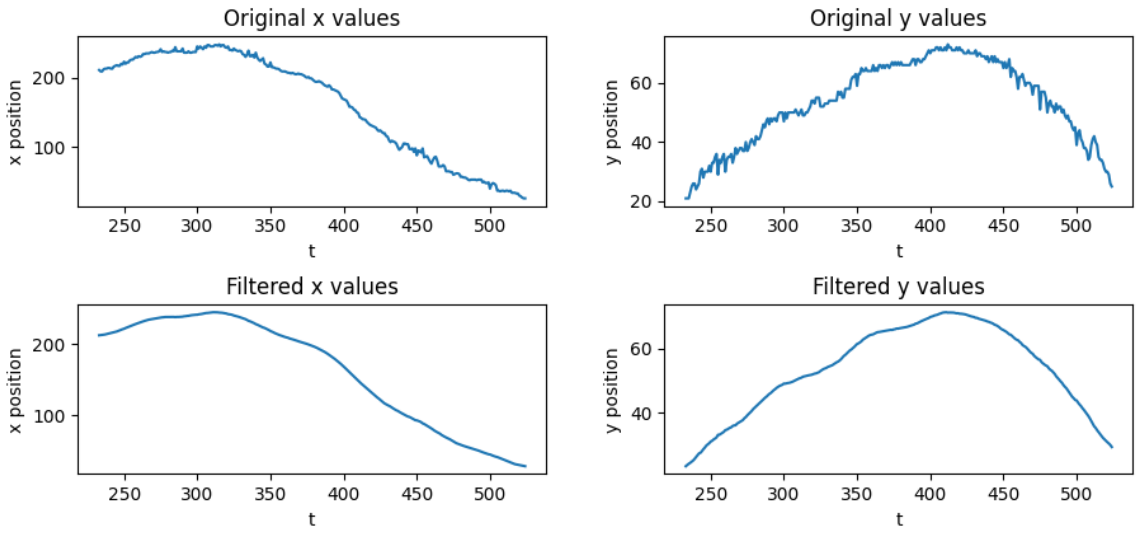

Figure 3.6: Filtering application on each position axis

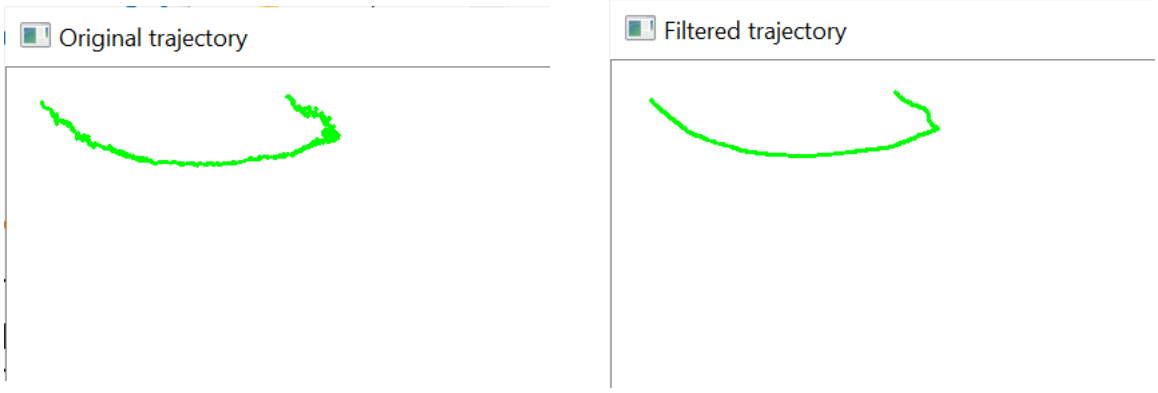

Figure 3.7: Filtering illustration: path before and after the filtering process

Segmentation

In the context of our domain, trajectory segmentation was a block that was essentially used as pre-processing to identify interesting episodes, so the different partitions could be classified instead of the whole trajectory. However, it can be very useful simply to divide the trajectory into smaller analysis pieces. As described in the implementation process, it is derived from the application of the Douglass Peucker algorithm regarding the position, speed, and angle time series of a given trajectory. Figure 3.8 illustrates a frame with the different trajectory segments drawn with a different color, resulting from the trajectory segmentation. Using this method, we probably won't cut the trajectory in the middle of some important moment. However, it is possible to verify that some of the segments have few points but this could be avoided by ignoring segments with small duration.

The epsilon parameter has to be tuned according to the data that is being used to implement these algorithms. This highly depends on the environment and the camera position/angle which will affect trajectories. Figure 3.9 shows different speed time series applying several epsilon values. As expected, a higher value will only trigger very important data points, as opposed to a low value that will output points that preserve better the original shape. In the context of its application, the idea is to not have unnecessary trajectory cuts, so non-conservative values were chosen to avoid a huge amount of segments. Table 3.1 complement this logic and illustrates the number of segments variation for different epsilons.

Figure 3.8: Trajectory segments using position epsilon=30, speed epsilon=2 and angle epsilon=50

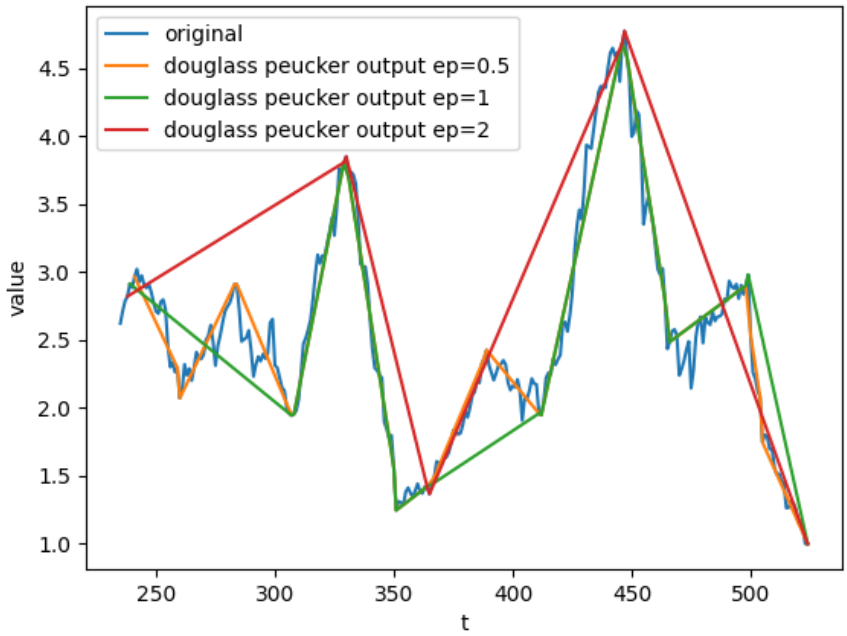

Figure 3.9: Douglass Peucker results on speed time series using different epsilon values

position epsilon	speed epsilon	angle epsilon	number of segments
10	0.5	10	55
10	0.5	50	38
10	2	10	38
10	2	50	20
30	0.5	10	52
30	0.5	50	35
30	2	10	36
30	2	50	18

Table 3.1: Number of resulting segments using different epsilon values

3.3.2 Feature Extraction

Most of the behaviors depend on the features that are extracted from the trajectory, so this package has a major role in the behavior detection task. Most of the features that are extracted are also described in the work in [18]. Overall there are features related to the motion, fish orientation, trajectory sparsity, and regions analysis. In total, the following features are calculated:

- velocity that can be calculated as $v = \frac{\Delta p}{\Delta t}$, and described as the position variation over time;
- acceleration that can be calculated as $a = \frac{\Delta v}{\Delta t}$ and described as the velocity variation over time;
- turning angle that can be calculated as $\theta = \arctan(p_t - p_{t-1})$, where $p_t - p_{t-1}$ is the motion vector;
- curvature that can be calculated as $k = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}}$, where x', y' and x'', y'' are the first and second derivatives respectively;

- centered distance that can be calculated as $cd = d(p_t, \mu)$, where p_t is the position on instant t, and μ is the mean position of the trajectory;
- normalized bounding box that can be calculated as $nbb = \frac{w}{h}$, where w and h are the width and height of the bounding box respectively.

Beyond the specified features, it is also registered information regarding the regions that the fish visits, and the transitions between those regions. It is possible to define as input a set of regions, and along with the trajectory processing, the counter for a given region is incremented every time the fish is detected on that region. Additionally, the same logic is applied when the fish cross to another region. Figure 3.10 illustrates an example of a trajectory with three defined regions. We can verify that the fish starts to being detected at the surface zone and meanwhile swims at the middle of the tank. Figure 3.11 complements this information with more detailed data: the example fish frequent the middle zone sensibly 75% of the time, and around 25% on the surface, while detected by the camera.

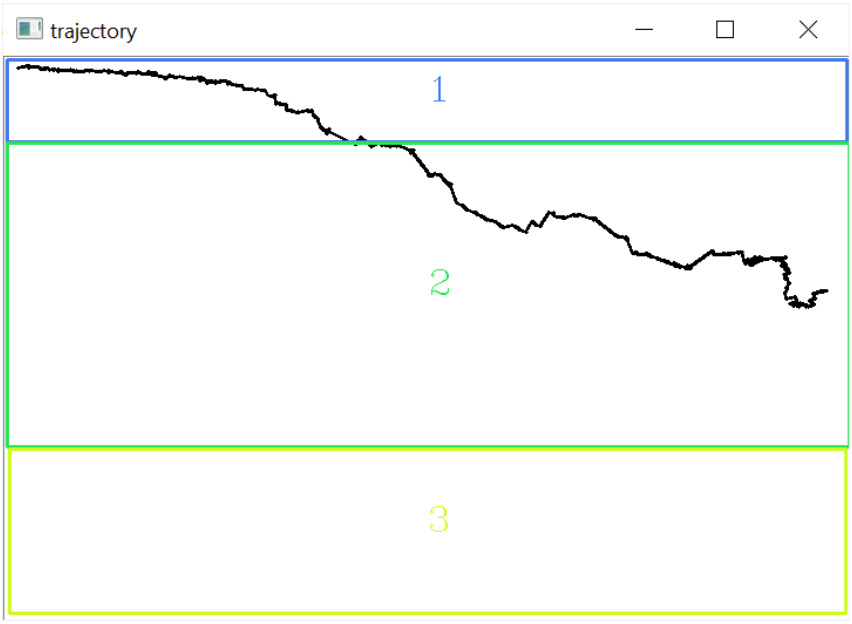

Figure 3.10: Example of a trajectory and specified regions

The enumerated features, with exception of the region-based features, can be calculated at every instant. This will afford to have a time series for each feature, such as the one draw in the Figure 3.12 which shows the normalized bounding box value along the trajectory lifetime. To represent a trajectory in a feature-vector format, several statistics are extracted from each time series and form a unique vector. The following statistics are calculated: mean, median, standard deviation, minimum, maximum, first quartile, third quartile, and finally the auto-correlation which can be seen as the correlation of the time series with a lagged representation of itself. Given that there are 6 features, and 8 calculated statistics,

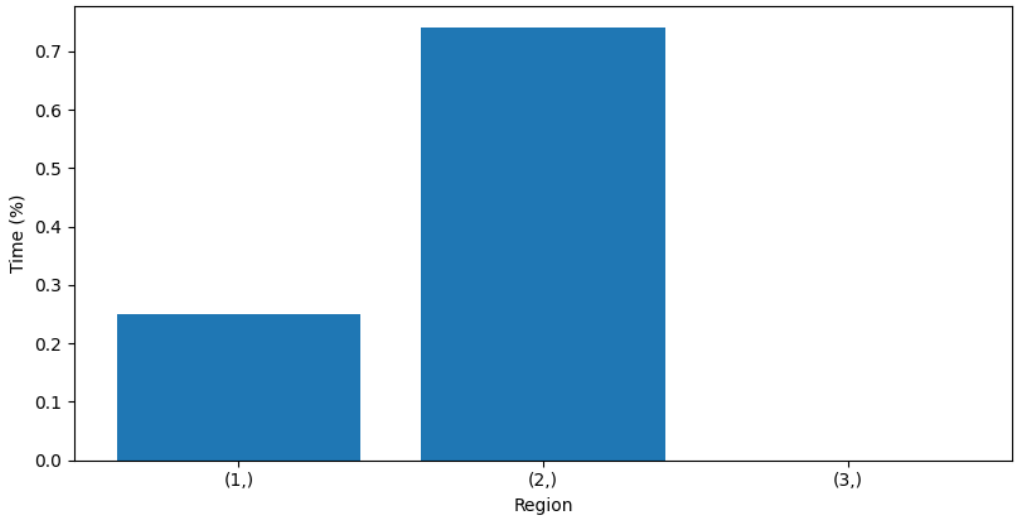

Figure 3.11: Regions frequency

and also the fact that the velocity and acceleration can also be separated by x-axis and y-axis, this gives a total of $10 * 8 = 80$ features. This vector can increase in size depending on the number of regions used. The frequency for each region is also added, as the number of transitions between each.

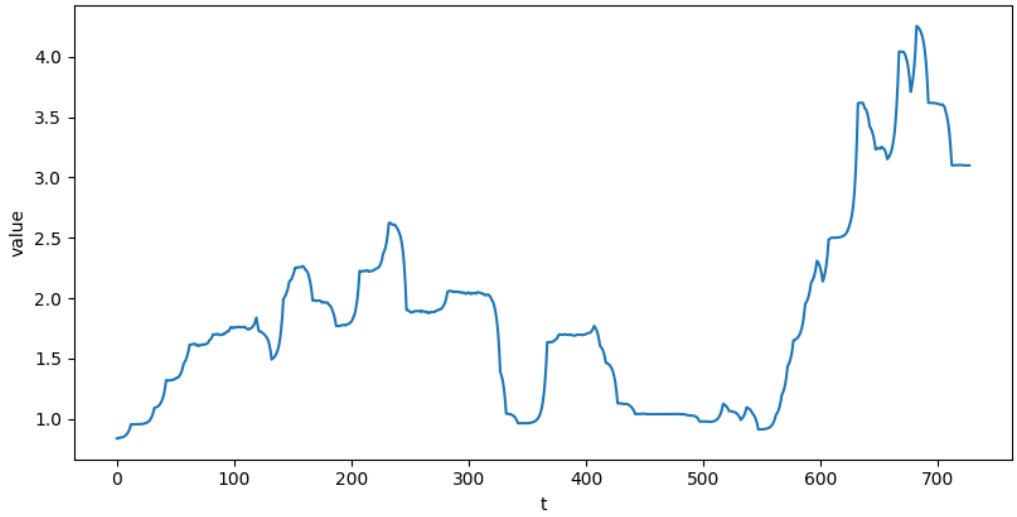

Figure 3.12: Time series of the normalized bounding box feature

Similar to what can happen on the trajectory path, and also from the same motifs, these time series can have noise that will impact statics that is calculated, such as minimum and maximum of each time series. If a given feature is extracted frame by frame, it is possible to verify some noise spikes. Those can be avoided if we increase the calculation period, for example, twelve by twelve frames. In this case, we can see that those spikes no longer appear (Figure 3.13). However, some data points are being wasted using this approach which is a downside. Another approach is applying, as in the trajectory

filtering, an exponential moving average filter. The results using different alphas are shown in Figure 3.14. We can see that several alpha values seem to be acceptable but a conservative value was usually used (0.01) to take into account the trajectories that have the most noise.

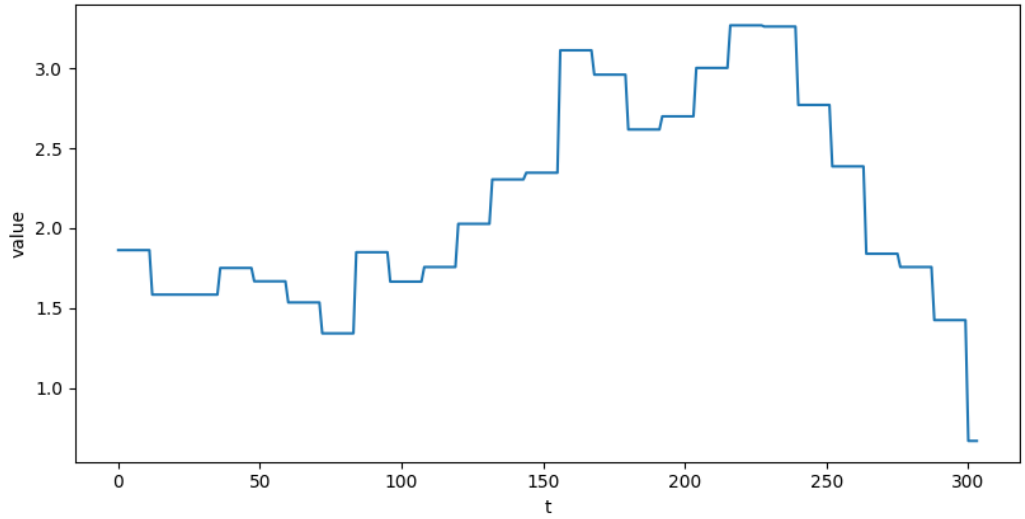

Figure 3.13: Velocity time series using a higher calculation period of twelve frames

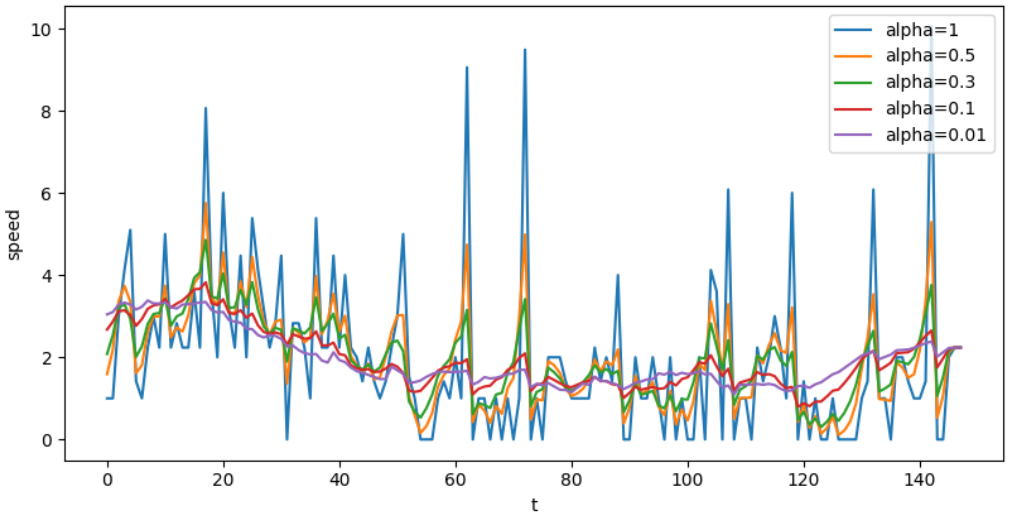

Figure 3.14: Exponential moving average using different alpha values

3.4 Anomaly Detection

3.4.1 Clustering-based

Clustering is a type of machine learning task whose goal is to group a set of samples. Despite being used to identify profile types and other similar domains, it is also known in the outlier detection field. This approach follows the logic also used in the work [18], but instead of using the positioning to group the trajectories, we use the features and the vector format described in the previous section.

First, all the trajectories are transformed into vector format and a clustering algorithm is applied using all these vectors. Looking at the output clusters, the groups that have few samples when compared with the total number of samples, are considered outlier groups. This logic is also described by the Figure 3.15. In this figure, we can verify that the red group only contains one sample, which is far away from the other cluster in green that contains six samples.

Two different clustering algorithms were applied: KMeans and DBScan. KMeans has the disadvantage of needing the specification of the number of clusters. To identify the outlier clusters we define a percentage threshold. Groups with less than $thr * \#samples$ samples are classified as outlier groups, where $\#samples$ is the total number of samples. On the other hand, DBScan has already the notion of outlier. It depends on two parameters: ϵ and $min\_samples$. If a given sample does not contain $min\_samples$ other samples within a distance ϵ, then it is classified as an outlier, so there is no need of specifying the percentage threshold.

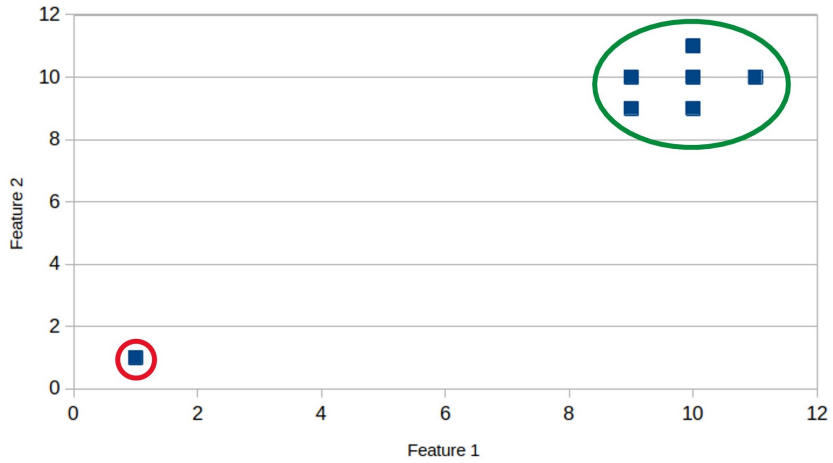

Figure 3.15: Clustering approach illustration

3.4.2 Switching Vector Model

The switching vector model was used to model pedestrian trajectories in work [2]. In this project, we try to use it to model the fish trajectories. In summary, it tries to model the different motion vectors for each image plane region, and also the transition between those fields. The image plane is described by a grid of nodes, where each region contains k motion vectors and a transition matrix. It is essentially characterized by the grid size, the number of fields, and two parameters that impact the fields update during training: δ which defines the neighborhood distance, and α that defines how attenuated distant vectors are.

To initialize the motion fields for each node, we resorted to a clustering algorithm. We extract all the motion vectors detected on that specific node and KMeans is applied on those, as we can verify in Figure 3.16. The red dots, the clustering centroids, are used as initial motion fields for this node. This logic is applied to every region in the grid. The training phase is based on the Expectation-Maximization algorithm which goal is to maximize the complete log-likelihood value. On every iteration, the motion vectors and the transition matrix for every region are updated according to this maximization function. Figure 3.17 shows the motion vectors for a given field in an initial state, and also after the training phase, and Figure 3.18 illustrates the complete log-likelihood value along the training process. As we can see, this value tends to increase which means that the input trajectories are better described by the updated parameters.

In the anomaly detection task, we can specify a minimum threshold for the joint probability. Given a new trajectory, the joint probability is calculated and if its value is lower than the specified threshold it is considered as an outlier.

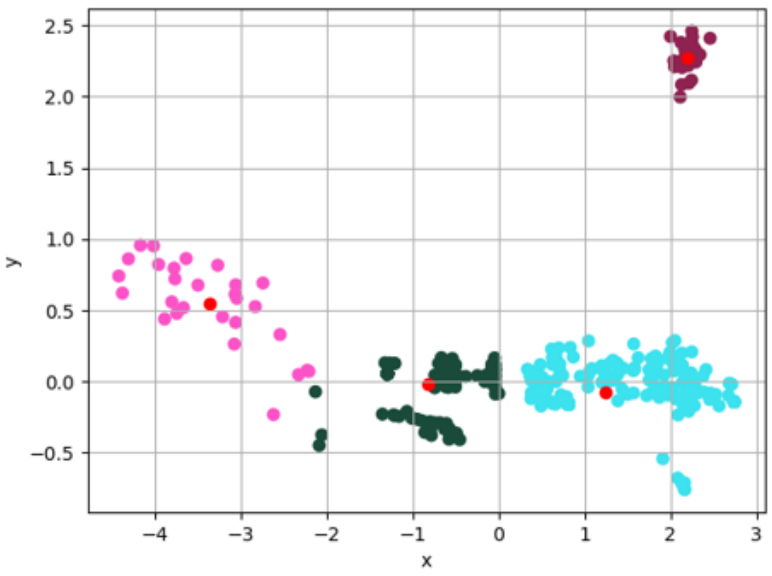

Figure 3.16: Vector fields initialization

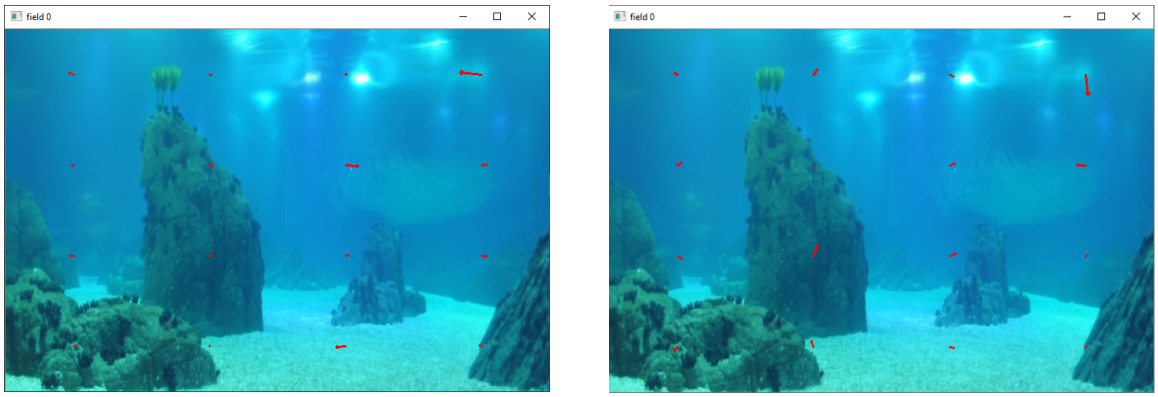

Figure 3.17: First field vectors before (left frame) and after training (right frame)

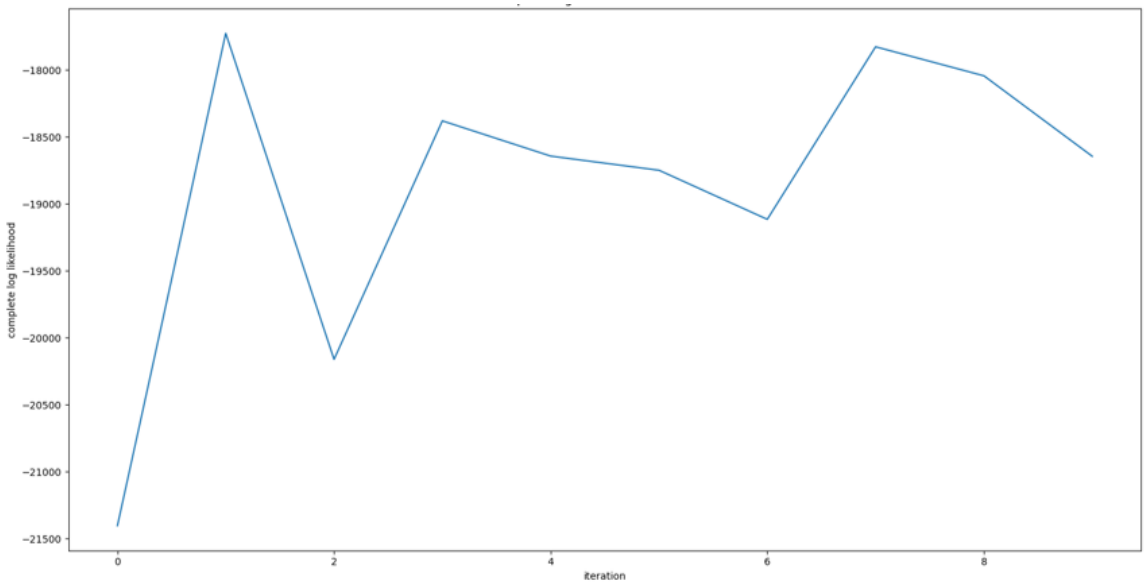

Figure 3.18: Complete log likelihood along the training iterations

3.5 Interesting Episodes Detection

3.5.1 Machine Learning Models

Interesting trajectories are trajectories that were marked with some interesting event, as described previously. We can visualize this problem as a binary classification task, where each trajectory can be classified as being interesting or not to be analyzed. Similar to the clustering strategy, this approach uses the feature extraction package to describe each trajectory in a vector format. Each vector has associated a binary label that is set to true if any event was notified during that trajectory.

Taking into account all the trajectories, these were transformed into a vector format and passed as input to a pre-processing pipeline before being transmitted to the model itself. The goal is to be able

to learn interesting trajectories feature patterns, to have the ability to generalize to other not known trajectory samples. Figure 3.19 complements this explanation. We can verify the three described phases: extraction of features from the trajectory, vectors pre-processing phase, and finally the model training phase.

Several models were experimented. We tried to explore the different classifier families: decision trees and random forest from a logic perspective, naive bayes from a probabilistic side, and support vector machines and k-nearest neighbors from a similarity view. Relatively to the pre-processing approaches, several methods were experimented to improve the model performance: normalization, class balance, principal component analysis, ANOVA feature selection, and correlation removal. Almost all these methods are already implemented, so we resorted to the *sklearn* library which is one of the known names on the machine learning libraries field.

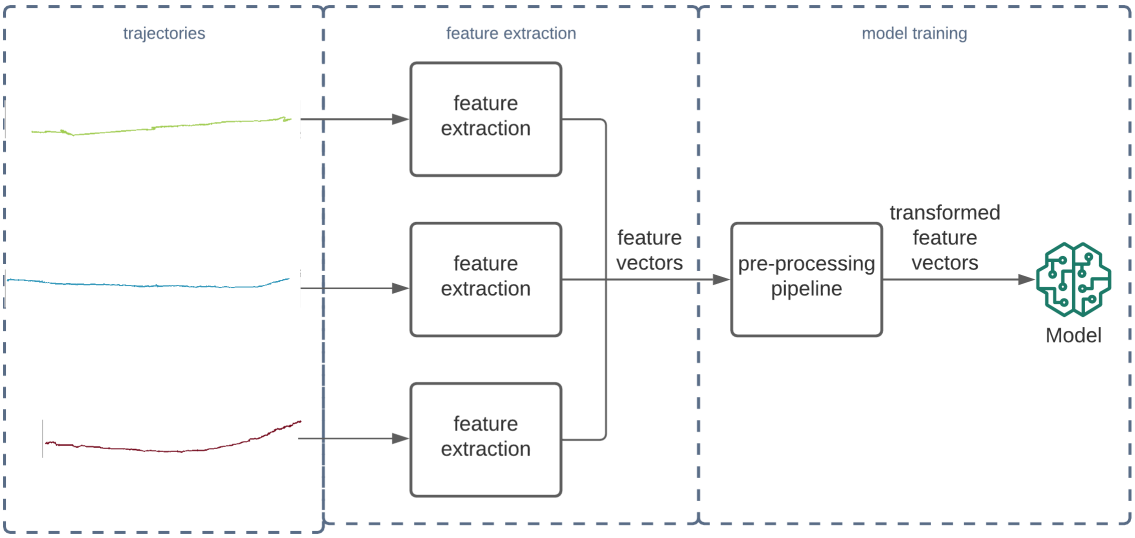

Figure 3.19: Machine learning model training process

3.5.2 Rule based

The ability to define rules is an alternative to the utilization of machine learning models. This approach has the advantage of being more user-friendly to understand the results. With this method, the user can define a set of rules regarding the extracted features, and also define the target species (sharks, manta rays, or others). The idea is to be able to define alert functions, but specific to a given species.

The rule validation is done through the feature time series that are extracted from the trajectory: velocity, acceleration, turning angle, curvature, normalized bounding box, and regions information. Each rule is defined by an interval of values and a minimum duration. The feature value has to be within the

specified interval for at least n consecutive frames, where n is the specified minimum duration. When this is detected, a highlight moment is saved. Figure 3.20 illustrates the block expected input and output, and Figure 3.21 shows an example of a highlighted moment detected on the turning angle feature for a given trajectory, using an interval value of 60 to 120, which reflects a significant ascent to the surface.

As an additional feature, it is also possible to specify a single threshold and use a minimum or maximum function. A single threshold is also what is needed relatively for region-based features. We can specify a maximum duration allowed for a given region. For example, manta rays do not usually frequent the bottom part of the tank, so we could specify a rule to trigger an event when that happens. The same logic can be applied to the transition between regions.

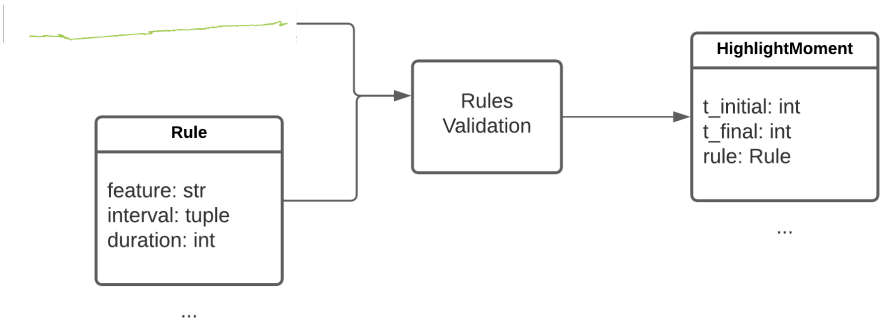

Figure 3.20: Rules validation block

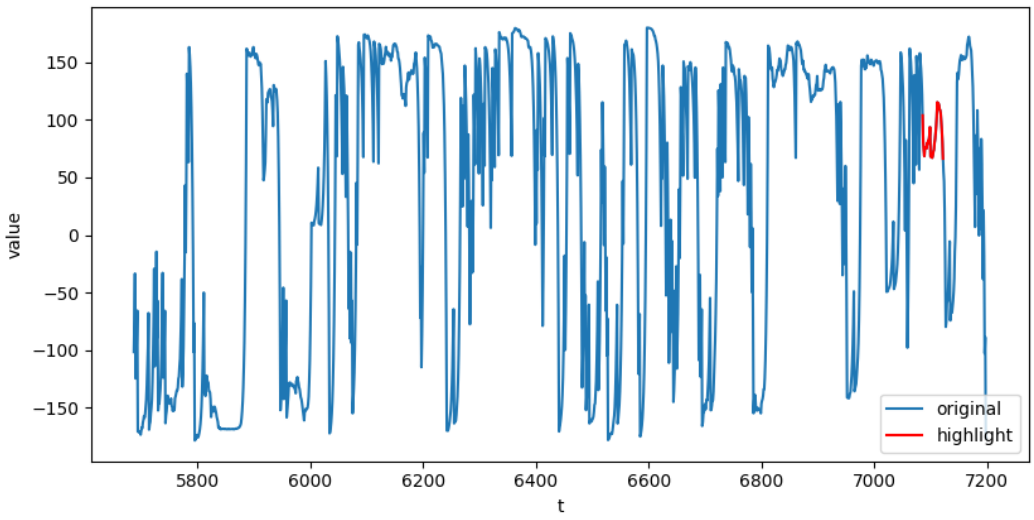

Figure 3.21: Highlight moment regarding turning angle feature

3.5.3 Switching Vector Model Adaptation

The switching vector model was also an approach implemented for the interesting moment's classification. The idea is to train two different models, so each model represents each of the classes: normal and interesting trajectories. Each of the sets is passed as input to a switching vector model, so this can learn suitable fields and transition information for each of the classes. After that, given a new trajectory, the joint probability is calculated using each of the models, and the argument associated with the model that gave a higher probability is the assigned class, as specified by

$$c = argmax[p(x|\theta_n), p(x|\theta_i)]$$

3.6 Feeding Period Detection

3.6.1 Convolutional Neural Network

Resorting on Convolutional Neural Networks (CNN) is one of the possible solutions to detect feeding periods. Using this type of model, the frame can be directly passed as input. Internally, a feature vector will be calculated using convolutional and pooling layers. This vector is then passed to a set of full connected layers to make the decision of being a feeding frame or not.

Several training videos were filmed for this behavior, so all frames from these videos can be used to train a model of this nature, so it can be able to learn image patterns for each of the classes. The work in [21] also uses a neural network, but with a sensibly different goal: measure the feeding hungry intensity. However, its architecture was used as baseline for this project.

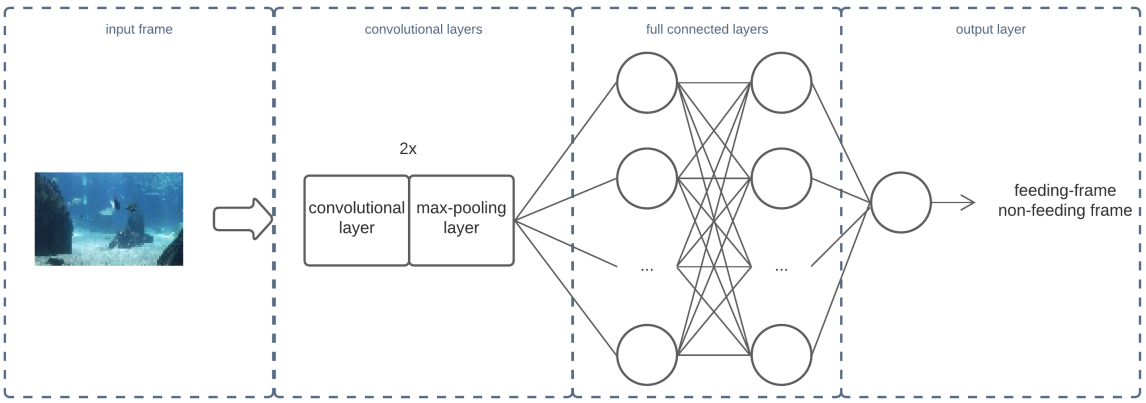

Figure 3.22: Baseline convolutional neural network for feeding frame classification

3.6.2 Motion-based Approaches

The feeding activity is characterized by the change on motion intensity around the feeding region. An alternative approach to the machine learning method is to be able to measure the amount of motion, and verify if this difference is visible between the feeding period and the normal period. We implemented two different ways of measuring the level of motion: apply consecutive frame subtraction and a threshold binarization to detect the number of active pixels (Figure 3.23); apply optical flow algorithm and calculate the average magnitude of the vectors (Figure 3.24).

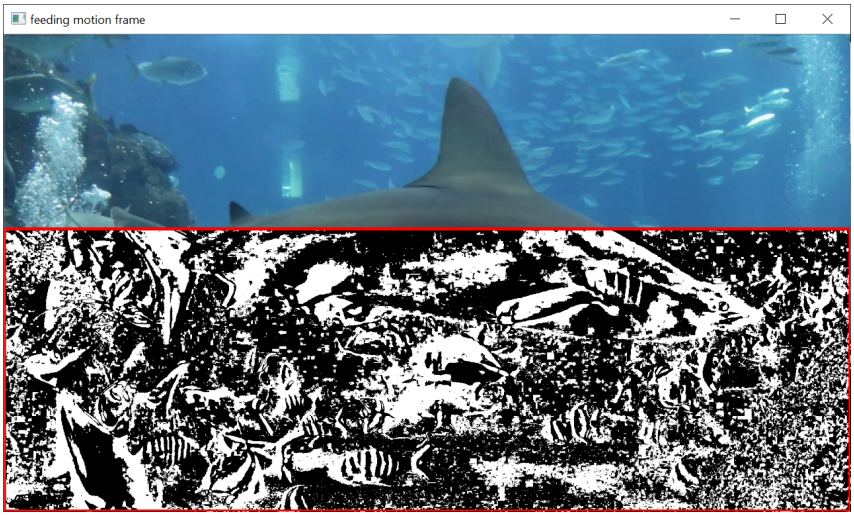

Figure 3.23: Active pixels in a frame during the feeding period and with a region defined

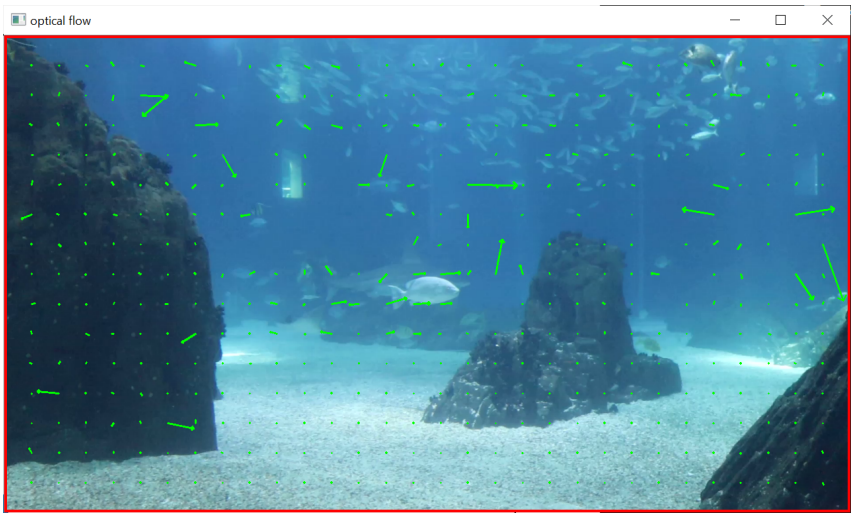

Figure 3.24: Optical flow result for an example frame

3.6.3 Aggregation-based Approach

Similar to the logic around the motion measurement, fishes also tend to aggregate more during feeding. To measure the level of aggregation of the detected fishes in a given frame we use the approach explained in work [8], which applies the delaunay triangulation on the fish's centroids that will form a triangular mesh between those points. However, we do not take into account fishes that seem to be far way from the center of mass (median position of all points). These outlier fishes are considered to not being interested on the feeding activity, which can be a motif of alert and a advantage of this approach. Figure 3.26 illustrates the resulting mesh and outlier from the application of this method on the frame of the Figure 3.25.

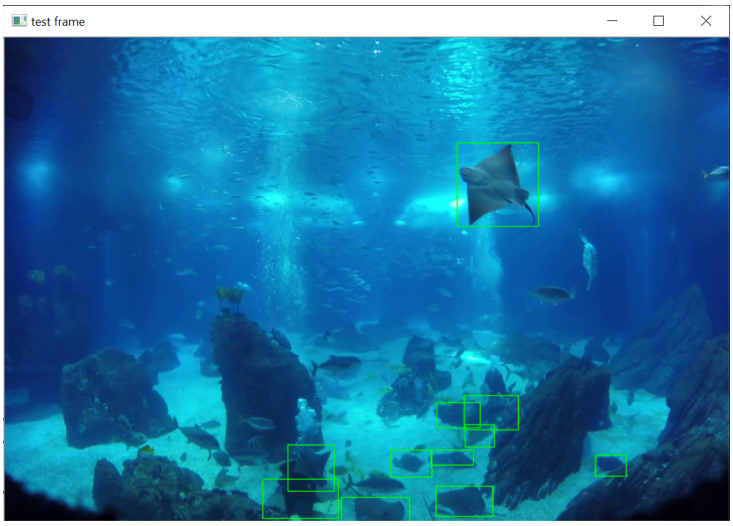

Figure 3.25: Frame during feeding and detected fishes

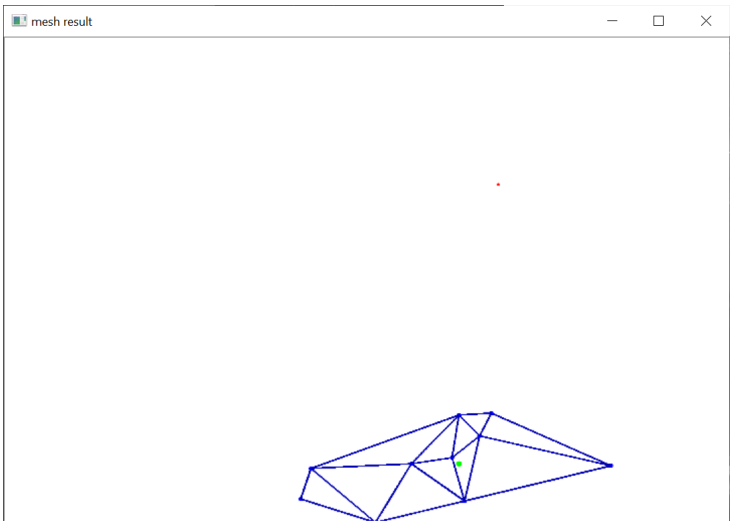

Figure 3.26: Output aggregation mesh

4

Evaluation

Contents

4.1 Datasets

Several data are needed to verify how the described algorithms work in the project domain. During the development of this project, many videos were filmed in the Lisbon Oceanarium. The focus tank was the main tank which has sharks, manta rays, and other species. These videos were filmed using a Canon camera and also, a go pro hero session 4.

Table 4.1 lists all the information related to the videos that were used in the experiments. All these videos are from the main tank of Lisbon Oceanarium. Notice that most of the videos have a high resolution and a considerable duration, which consequently justifies a high video size (300MB to almost 10GB).

video name	resolution	duration (min)	fps	frames	behavior usage
v29	720x480	5:00	24	7200	anomalies/interesting
feeding-v1-trim	1920x1080	5:02	30	9060	feeding (bottom feeding period)
feeding-v1-trim2	1920x1080	5:00	30	9000	feeding (bottom period without feeding)
feeding-v2	1920x1080	5:52	50	17600	feeding (bottom feeding period)
feeding-v3	1920x1080	22:09	50	66450	feeding (surface feeding period)
GP011844_Trim	1920x1440	3:04	30	5520	feeding (bottom feeding period using go pro)

Table 4.1: Information about the videos used for evaluation

Video 29 came from the previous project, and preliminary ground truth of some fish's motion was already available. The rest of the videos are related to the feeding behavior, once they do not need fish detection data nor tracking. With this project, video 29 ground truth was enlarged with more fish trajectories, from all detected sharks and manta rays which are the main focus species of this project. To do so, we resorted to the Darklabel application. All the detected fishes were saved into a JSON format file, that facilitates the parse of this data directly to a suitable format every time this data is needed. It can be described as a dictionary where each fish corresponds to a new entry. Each entry value is also a dictionary with two entries: the trajectory and the bounding boxes of the associated fish. The following file portion complements the described structure.

```
1 {
2     "<fish-id>": {
3         "trajectory": [[<t0>, <x0>, <y0>], ..., [<tn>, <xn>, <yn>]],
```

```
4           "bounding-boxes": {
5               "<t0>": [<width>, <height>],
6               ...,
7               "<tn>": [<width>, <height>]
8           }
9       },
10      ...
11  }
```

Data exploration was performed regarding the trajectories data. Two interesting points to be analyzed are the trajectory duration and the most frequented regions for each of the species. Regarding duration, most of the trajectories have a value of up to twenty seconds but there are samples with higher values. Figure 4.1 illustrates the duration's histogram for the sharks, and confirms the specified values: cases that have a duration higher than twenty seconds are rare. The scenario for the manta rays is similar.

Relatively to the regions, we could verify that the manta rays frequent less the bottom part of the tank, which consolidates the information given by the biologists. Positions heatmap of the manta rays species is drawn by Figure 4.2. Each cell on the heatmap represents the number of positions detected on that specific region for all trajectories. We can see that the bottom part has mainly zero positions detected, as opposed to the top part.

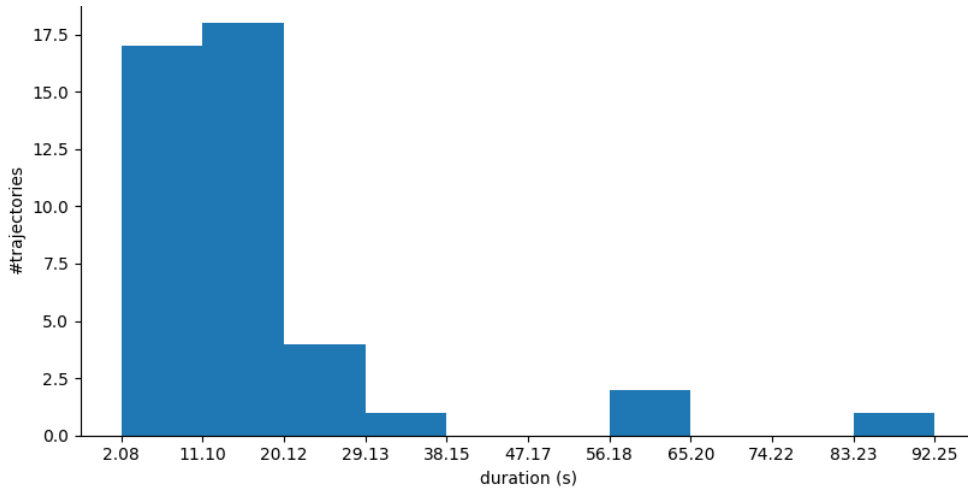

Figure 4.1: Trajectories duration histogram for sharks

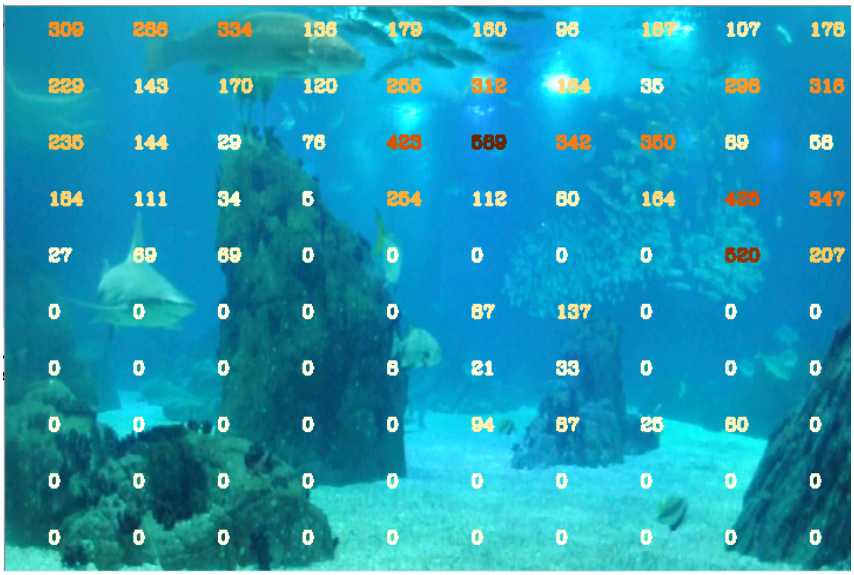

Figure 4.2: Regions frequency for manta rays

4.1.1 Trajectories vector-format dataset

To avoid repeating the feature extraction phase for all the trajectories, a dataset was created in a feature vector format. To this purpose, the feature extraction package was used. As previously explained, several features are extracted from the trajectory: speed and acceleration (for both dimensions, x-only axis, and y-only axis), turning angle, curvature, distance to center, regions frequency, and regions transition. For each of these time series several statistics are extracted: mean, median, standard deviation, minimum, maximum, first quartile, third quartile, and auto-correlation. This gives a total of eighty nine features. A trajectory can be represented by a vector of the specified shape, instead of a sequence of position points. The dataset building can be summarized by:

1. reading fishes from the fishes ground truth file
2. apply pre-processing: fill the gaps, and smooth the trajectory
3. extract the features for each of the trajectories
4. save all the trajectory vectors to the dataset file: comma-separated values file where each row is a different trajectory vector sample

Correlation is most of the times the main concern on the datasets. Two variables are correlated if they move in coordination with one another. In other words, if two variables have a high correlation, one is easily calculated by knowing the value of the other. This means that one of them is probably expendable, and it is only contributing to the high dimensionality which can impact models performance (curse of dimensionality). Taking into account all the vectors, the correlation matrix was calculated and

the histogram of those values was visualized (Figure 4.3). As we can see, most of the correlation values are not high. On Figure 4.4, we can also verify how the dimensionality varies when deleting one of correlated features, considering a correlation relationship two variables that correlate higher than a given threshold. When using a threshold of 0.9 we can decrease the dimensionality value to approximately 80 features, and when using 0.8 it is possible to decrease sensibly another 10 features.

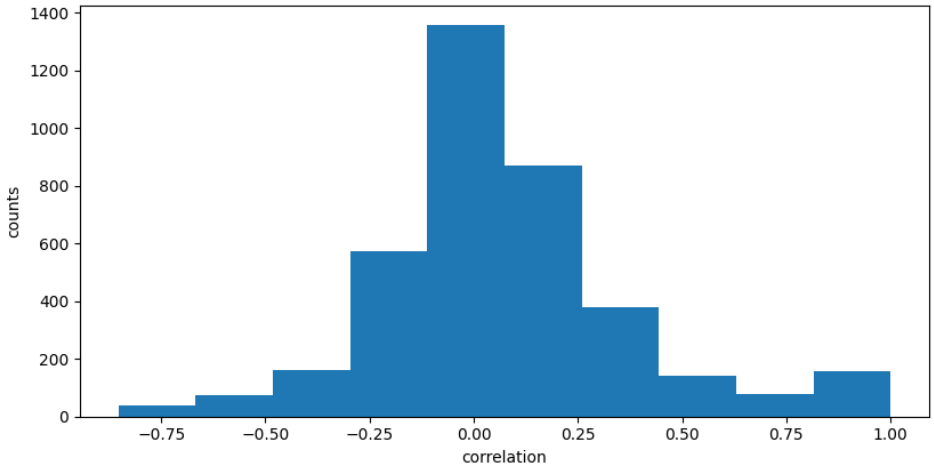

Figure 4.3: Histogram of the correlation matrix values

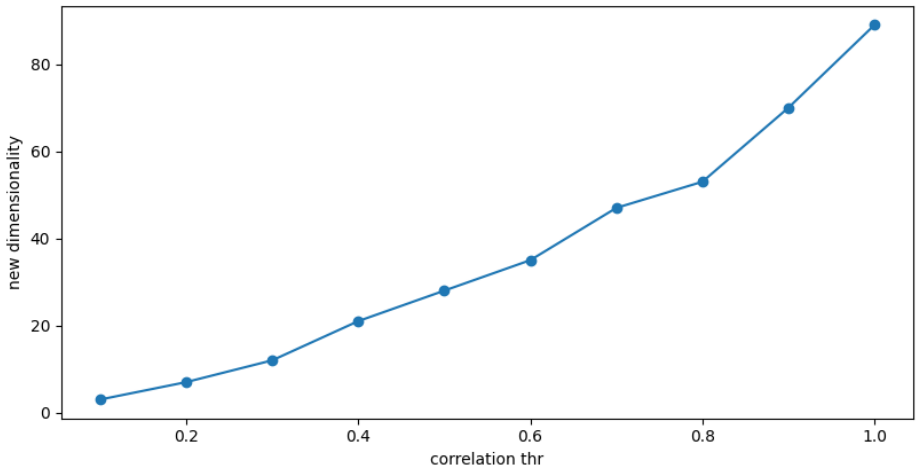

Figure 4.4: Dimensionality variation using different correlation thresholds

The probability density plots were also analyzed for the different features. Figure 4.5 shows an example regarding the median of the speed. Each of the lines represents the values for a different species. Tuna incorporate all the fishes that are not sharks nor manta rays. Regarding this feature, we can conclude that they have the highest speed. On the other hand, sharks and manta rays look similar, but sharks have a slightly higher value.

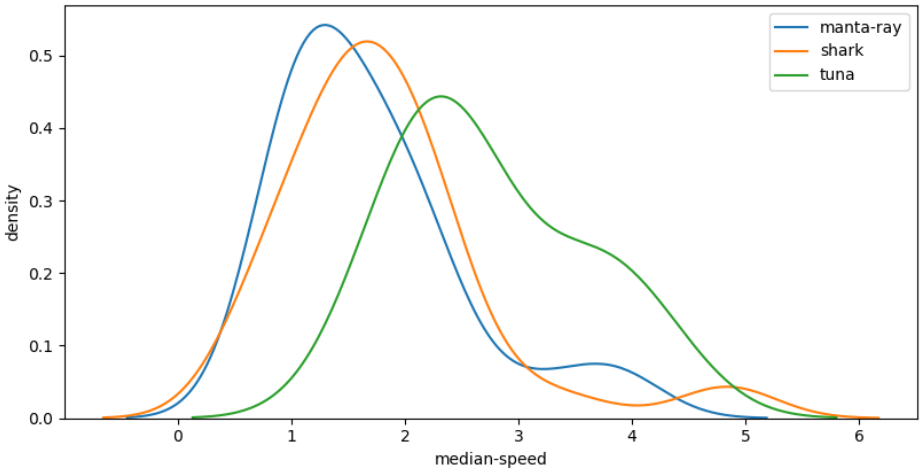

Figure 4.5: Probability density plots for the median of the speed feature

4.1.2 Feeding datasets

Most of the used videos are related to feeding. One of the reasons is the fact that the feeding detection approaches do not rely on trajectories data, so ground truth is not needed. Additionally, the feeding behavior is the most visible to the human eye, when compared with anomalies and interesting moments which are more subjective. Datasets regarding this behavior were built, for both feeding regions (bottom and surface) and also using a different perspective (GoPro).

Table 4.2 specifies information about these datasets, including the videos used to build each one, the total number of frames, the training/testing division, and the percentage of feeding frames. The number of total frames and the number of feeding frames are consequently related to the video size and frames per second, and how much time was the feeding period filmed, respectively. In every scenario, an 80-20 splitting percentage was used. While making this division, we make sure that consecutive frames are not in separated sets, to avoid having extremely similar frames in the train and test set. Also, all the frames were saved with a significantly low resolution (80x50), given its usage as input for convolutional neural networks.

dataset	videos	total frames	training set	feeding frames
bottom (cannon)	feeding-v1-trim feeding-v1-trim2 feeding-v2	35686	80%	57%
bottom (gopro)	feeding-v3	21600	80%	31%
surface (cannon)	feeding-v4	66474	80%	61%

Table 4.2: Information of feeding datasets

Figures 4.8, 4.11, and 4.14 shows an example frame for each of the states (feeding and non-feeding),

and also for the different datasets: bottom-feeding using the cannon camera, bottom-feeding using the GoPro camera, and surface feeding using the cannon camera, respectively. We can confirm the motion variation on the bottom region when comparing the feeding frame and the non-feeding frame of the bottom datasets. However, the feeding at the surface is not so visible, even for the human eye. A set of sticks are used (highlighted by a red circle in Figure **??**), but there are not many and they are thin which difficult its visibility, even when using a high resolution.

Figure 4.6: feeding frame

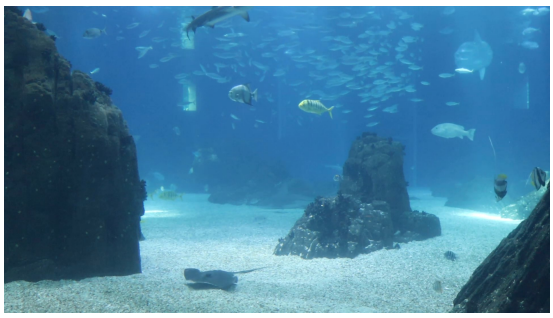

Figure 4.7: non-feeding frame

Figure 4.8: Example frames for bottom feeding using the Cannon camera

Figure 4.9: feeding frame

Figure 4.10: non-feeding frame

Figure 4.11: Example frames for bottom feeding using the gopro camera

4.2 Experiment A: Anomaly detection DBScan vs KMeans

Clustering, as explained before, can be useful to identify trajectories that do not follow the usual trajectory features (outlier trajectories), and also to discover groups of patterns among the trajectories. This experiment has the goal of evaluating the quality of the anomalies identified by the clustering algorithms: DBScan and KMeans. An anomaly can be defined as something that deviates from it is standard, so

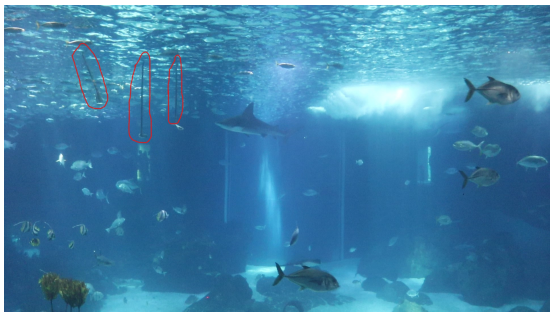

Figure 4.12: feeding frame

Figure 4.13: non-feeding frame

Figure 4.14: Example frames for surface feeding

trajectories that are far away from the general group of trajectories can be classified as an anomaly.

In this experiment, several clustering pipelines are applied in the trajectories dataset and the silhouette metric is extracted to get an idea of the quality of the output groups. Pre-processing methods can play an important role, especially in the machine learning field. Table 4.3 summarizes the pre-processing methods, in terms of their functionality and presents also the short formula for each one.

method	**description**	**formula**
normalization (standard scaler)	vector features can have different scales and this can have an impact specially on models that regard on distance and similarity, which is the case of these clustering approaches	$z = \frac{x-\mu}{\sigma}$
principal components analysis (PCA)	projection of the data on the axis with highest variance, helping highlighting differences between the groups. W are the eigenvectors	$y = W^T X$
feature selection (ANOVA f-value)	ratio that take into account the uni variate variance within groups and between groups, we can use this ratio to select the K features with higher value	$F = \frac{\text{between groups}}{\text{within groups}}$
correlation removal	when two variables have a correlation value higher than a given threshold, one of them is removed. s_{xy} is the covariance	$r_{xy} = \frac{s_{xy}}{s_x s_y}$

Table 4.3: Pre-processing methods summary

In terms of algorithms, DBScan is a density-based clustering approach. For this reason, it can automatically identify outlier samples. It relies essentially on two parameters: epsilon (distance threshold), and the minimum samples. If a given sample does not have the minimum samples within the epsilon distance, it will be classified as an outlier. On the other hand, KMeans needs the number of cluster specifications (parameter k), and the distance metric can also have an impact on the results. Given that the number of clusters needs to be specified apriori, as explained before, a cluster is classified as an outlier if it has few samples when compared with the total number of samples.

Regarding metrics, the silhouette is usually the internal index used. It does not regard external

information and takes into account the cohesion and separation of the output clusters.

Cohesion reflects how close the samples of a given cluster are to each other. It is the mean distance between a given sample and the other samples of the same cluster and can be calculated as

$$cohesion(i) = \frac{1}{|C_i| - 1} \sum_{j \epsilon C_i, i \neq j} d(i, j), \tag{4.1}$$

where C_i is the cluster on which the sample i was assigned. On the other hand, separation represents how separate the output clusters are from each other. Separation of a given sample is the minimum mean distance found between that sample and all samples of another cluster, and it can be calculated as

$$separation(i) = \min_{k \neq i} \frac{1}{C_k} \sum_{j \epsilon C_k} d(i, j), \tag{4.2}$$

where C_k is a cluster that does not contain the instance i. Finally, the silhouette combines these two metrics, and it is calculated as

$$silhouette(i) = 1 - \frac{cohesion(i)}{separation(i)} \tag{4.3}$$

An ideal silhouette is a value close to 1. In this case, the separation value is significantly higher than the cohesion value. The quality of the output trajectory clusters can be measured by these metrics, and consequently, the quality of the detected outliers.

4.3 Experiment B: Interesting episodes detection

As explained before, there is a set of criteria to define interesting moments. It can also be seen as an abstraction of anomalies. An anomaly is always an interesting moment to be analyzed. Given these criteria, interesting moments were identified, by hand, on the sharks and manta rays trajectories of video 29, and saved into a ground truth file. A given trajectory is considered to be interesting if there is any episode marked on it.

The goal of this experiment is to evaluate the capability of classifying correctly these trajectories as being interesting, or not, to be analyzed. To do so, several models were trained and evaluated, using also different pipelines. Table 4.4 summarizes the models that were experimented. Classifiers from different families were evaluated. Regarding parameters, a grid search was performed to discover the most suitable set of parameters for each classifier. Notice that decision trees and random forests have more interesting parameters to be checked due to the tree learning process. Relatively to pre-processing, the same pipelines were experimented with as in the clustering approaches (Table 4.3).

model	family	main parameters
K-Nearest Neighbor	similarity	n_neighbors, metric
Support Vector Machine	similarity	C, kernel, degree, gamma
Decision Tree	logic	criterion, min_samples_split, min_samples_leaft, min_impurity_decrease
Random Forest	logic	n_estimators, criterion, max_depth, max_features, min_samples_leaft, min_samples_split
Naive Bayes	probabilities	-

Table 4.4: Machine learning models summary

Holdout technique was the chosen evaluation approach, given the number of samples of the dataset. This method consists of training the model with all the samples except the one that is pretended to be evaluated, and this process is repeated for all the samples. This way we can guarantee that the classifier is not trained with few samples to be evaluated. In terms of evaluation metrics, the traditional precision, recall and accuracy were calculated: equations 4.4, 4.5, and 4.6 respectively. A true positive(TP) sample is an interesting trajectory that was classified as interesting, and in another hand, a true negative(TN) is a trajectory without any marked episode that was classified as not being interesting. Additionally, the precision-recall curve was visualized and the area under the curve was calculated. These can be especially useful to verify the capability of the model to react to different decision thresholds, given that there are problems that precision can be more important than recall or vice versa.

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.6}$$

Regarding this task, three additional experiments were tested. The first one was related to the species. The idea was to train a model per species (one for sharks and another for manta rays), instead of using all the samples in just one generic model, and verify if better results are achieved. The second hypothesis tried to take advantage of segmentation and interesting moments characteristics. These moments are characterized by having a short period. The goal was to verify if better results are obtained if the trajectory segments are classified instead of classifying the trajectory as a whole.

Finally, the last experience adapted the switching vector model to the interesting episodes classification task. In this adaptation, two vector models were trained: one using as input the trajectories considered to be interesting and the other using the trajectories without any marked episode. Given a

new trajectory, the joint probability is calculated regarding these two models, and the one that gives a higher value is the class assigned. This model is essentially characterized by four parameters: number of nodes, number of fields, delta, and alpha. Several models were trained using different parameter values and the complete log-likelihood was calculated for each one. The joint probability reflects how well the model describes the trajectory passed as input and it can be calculated as

$$p(x, k|\theta) = p(x_1, k_1) \prod_{t=2}^{L} p(x_t|k_t, x_{t-1}) p(k_t|k_{t-1}, x_{t-1}), \tag{4.7}$$

where x is a sequence of positions of a given trajectory, k its active vectors, θ is the model itself, and L is the length of the trajectory. The complete log-likelihood is the sum of the logarithm joint probability for all the trajectories, and it can be calculated as

$$p(\chi, \kappa|\theta) = \sum_{j}^{S} \log p(\chi_j, \kappa_j|\theta), \tag{4.8}$$

where χ is the set of trajectories, κ is the set of active vectors for each trajectory, and S is the number of trajectories. The models that gave a higher complete likelihood value were the ones evaluated for this task. For the evaluation purpose, the same metrics were used as the ones used regarding machine learning models: precision, recall, and accuracy.

4.4 Experiment C: Feeding Period Detection CNN vs Motion Based methods

In the set of videos that were filmed, there are several regarding feeding activity. Sharks are fed near the surface using sticks with food on the tip, and some of the manta rays are fed on the bottom of the tank by divers. The goal of this experiment is to evaluate the performance of the different approaches for detecting feeding periods. The following list summarizes the methods that were evaluated:

- convolutional neural network(CNN): the frame is passed as input to the network, and it is classified as begin a feeding frame or not;
- active pixels: motion is determined on each instant based on consecutive frames subtraction, and a motion threshold is applied to classify it as being a feeding frame or not;
- optical flow: the motion vector is calculated for a grid of points, and a motion threshold is applied to the average motion to classify it as being a feeding frame or not.

In this context, a true positive(TP) is a feeding frame that was classified as being part of the feeding activity. On the other hand, a true negative(TN) is a frame that is not part of the feeding period, and it

was assigned to the non-feeding class. Given this logic, precision, recall, and accuracy were calculated to evaluate each of the approaches (equations 4.4, 4.5, and 4.6 respectively). Additionally, the timeline of the errors was also visualized, to verify if most of the errors are near the feeding period switch.

CNN parameters can have a significant impact on the performance, either in terms of architecture or in terms of hyper-parameters. Tables 4.5 and 4.6 contain a summary of the set of parameters that were tuned. A grid search was applied on a different validation set to verify the most suitable values for each one.

Regarding the motion methods, they regard essentially on the threshold definition. To choose this parameter, the motion time series was visualized for a given training video, and a value that divides the feeding frames motion from non-feeding frames motion was chosen. Additionally, these methods were also performed using the region definition feature, so the motion on the other regions does not interfere with the decision. For example, if we are interested in detecting the bottom-feeding activity, we could exclude the detected motion on the upper side of the tank.

parameter	**description**
learning rate	step size at each iteration
dropout rate	percentage of neurons that are ignored
activation function	defines the output of a neuron
error function	measure of prediction

Table 4.5: Hyper-parameters description

parameter	**description**
number of convolutional layers	layer that applies a set of filters
number of hidden layers	layer of mathematical functions that produce a given output
number of neurons per hidden layer	mathematical function units
number of filters	matrices that slide over the image (convolution)

Table 4.6: Architectural parameters description

5

Results

Contents

5.1 Results of experiment A: Anomaly detection DBScan vs KMeans

One of the main approaches to detect anomalies is based on clustering. Clusters with few samples can be seen as a set of anomalies, its features deviate from the general group of samples. Two algorithms have been experimented with: DBScan and KMeans.

To choose the best pipeline to be used with DBScan, a set of experiences using different pre-processing methods was made. Figure 5.1 represents the pipeline that gave a better performance considering silhouette metric: a standard scaler as the normalizer, followed by a feature selector using the ANOVA f-value and finally applying the DBScan cluster algorithm. Resorting on this pipeline, a silhouette of nearly 0.72 was achieved which was significantly higher than the ones obtained utilizing the other pipelines (Table 5.1). Only one outlier was detected from the set of trajectories, as we can see in Figure 5.3 and 5.4: a shark that does a considerable curvature to the bottom and suddenly changes direction turning back to the origin point of detection. However, if we relax the epsilon value, more samples can be seen as outliers but the silhouette value also decreases, as we can verify in Figure 5.2.

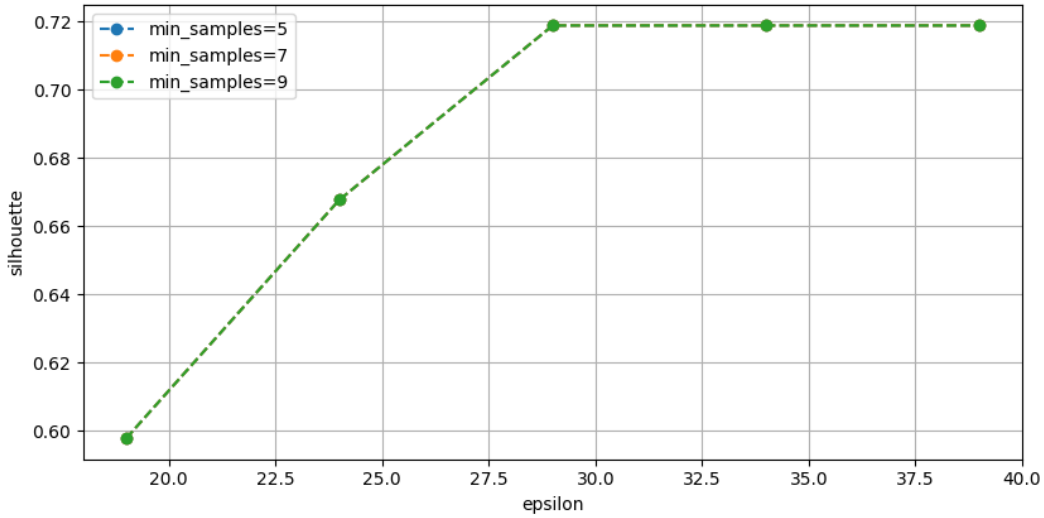

Figure 5.1: Best DBScan pipeline: normalizer + feature selection + DBScan

Pipeline	epsilon	min-samples	distance metric	silhouette
DBScan	106	9	euclidean	0.48
Normalizer + DBScan	102	5	manhattan	0.45
Normalizer + PCA + DBScan	11	5	euclidean	0.51
Normalizer + FS + DBScan	29	5	manhattan	0.72
Normalizer + CVR + DBScan	80	5	manhattan	0.46

Table 5.1: DBScan pipelines' results

The DBScan algorithm has the advantage of not being necessary to specify the number of clusters, unlike the KMeans algorithm. In Figure 5.5, it is possible to visualize the cohesion value calculated from

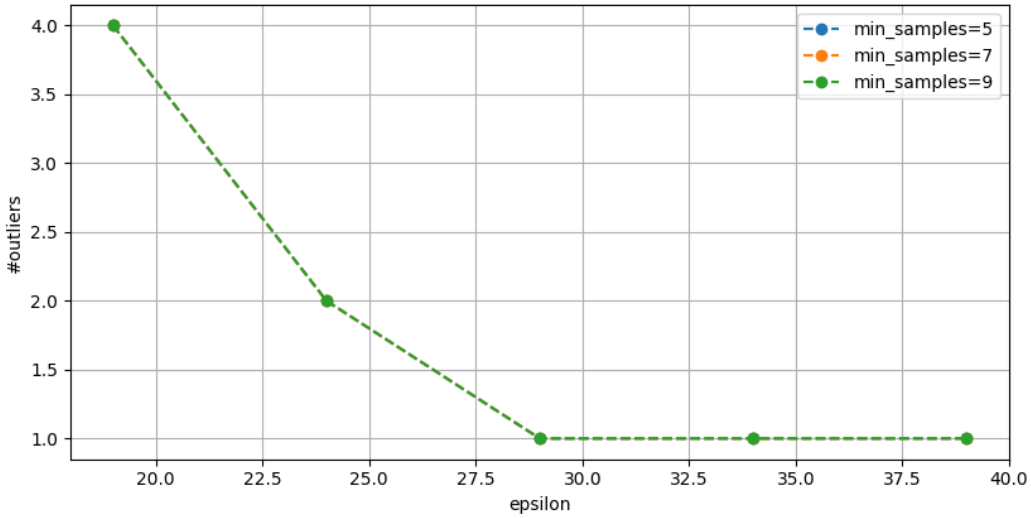

Figure 5.2: Number of detected outliers using different DBScan parameters

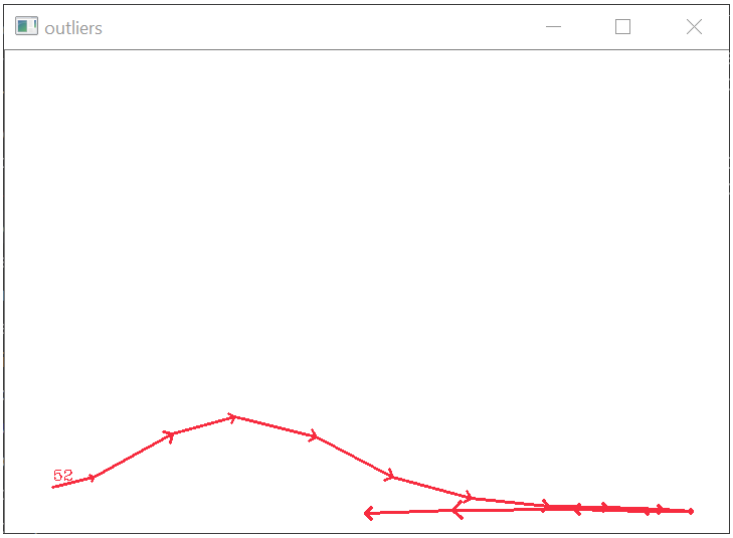

Figure 5.3: Identified outliers using the best DBScan pipeline

the resulting clusters, using different numbers of clusters (k value). Although it is not possible to observe a standing out an elbow, we can verify that from a value of approximately seven clusters the cohesion starts decreasing at a slower pace.

Looking at the distribution of the resulting clusters (Figure 5.6), and applying the outlier logic, we can consider clusters 0/2/4 as outliers because they have considerably fewer samples when comparing with the rest of the clusters, such as cluster 2 which is the major one. It is possible to visualize the trajectories from clusters 0 and 2 in Figure 5.7. As we can see, there is a very pronounced pattern in each of the clusters. Group 0 tends to keep the same direction (constant turning angle) during all the trajectory, and although it is not represented in the image they also have similar x-axis speed values. This can

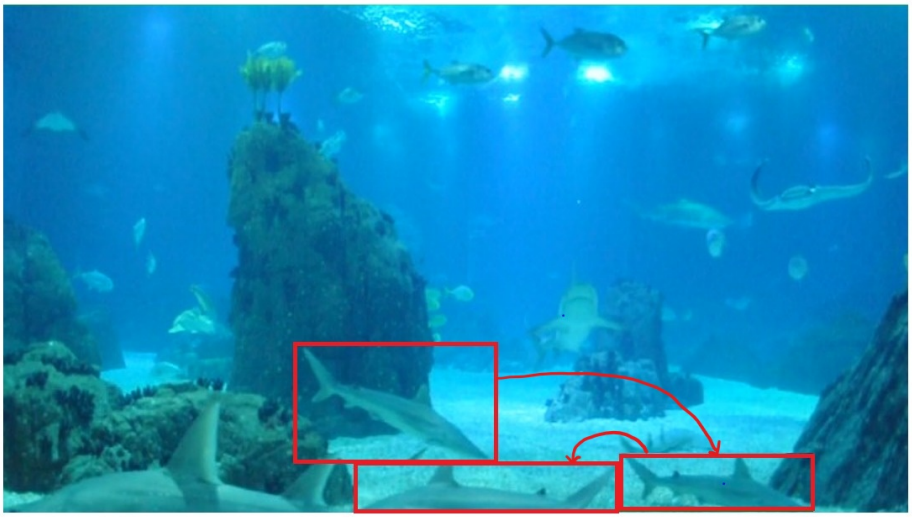

Figure 5.4: Sequence of frames of the detected outlier

also be concluded from Figure 5.8, which presents the set of features that most differentiate from the other clusters. Group 2 contains the identified outlier using the DBScan algorithm. It also contains two additional trajectories which have the same motion behavior: a sudden direction change turning back to the origin of the trajectory. Although these resulting clusters seem to represent interesting groups according to motion patterns, the overall performance (silhouette) value was sensibly 0.30, thus we believe that the DBScan algorithm is more appropriated for the outlier detection task.

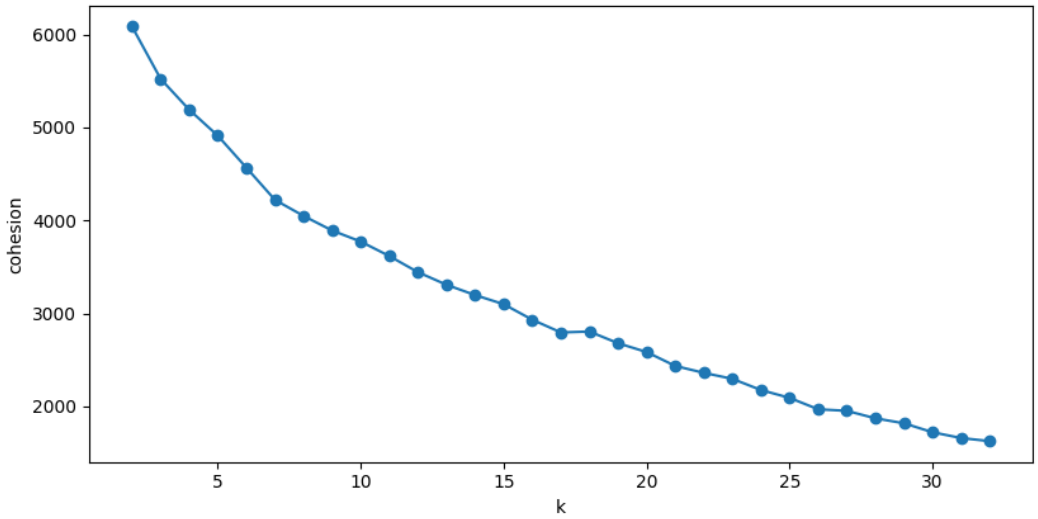

Figure 5.5: Cohesion value for different number of clusters

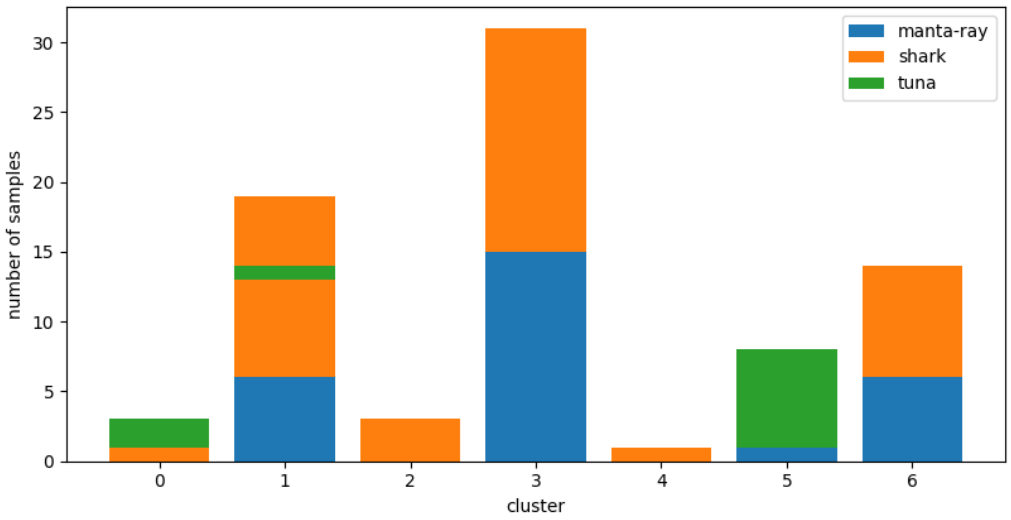

Figure 5.6: Clusters distribution using KMeans

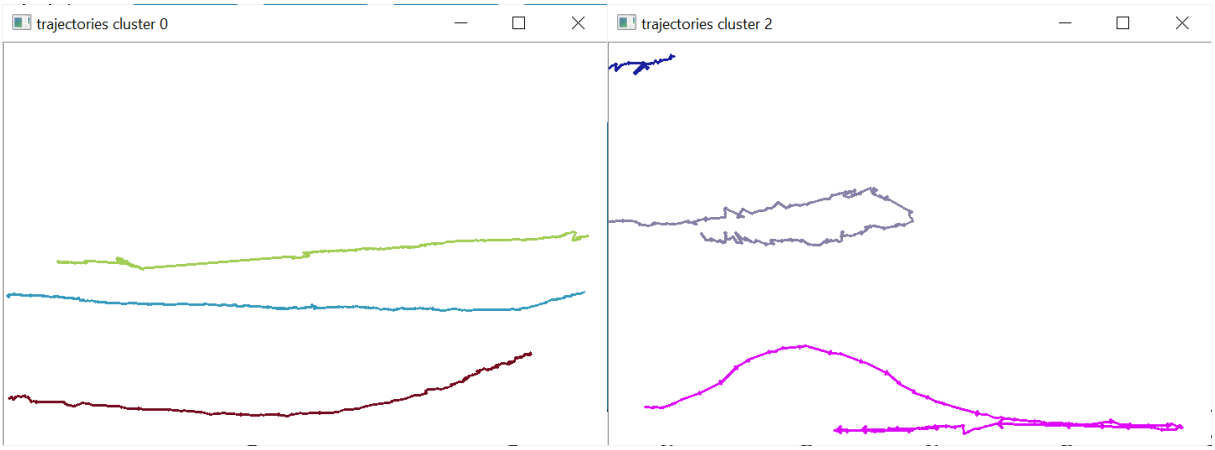

Figure 5.7: Resulting trajectories from cluster 0 and 2 respectively

5.2 Results of experiment B: Interesting episodes detection

5.2.1 Machine Learning models

Interesting episodes are trajectories, or segments of trajectories, that are considered interesting to be analyzed by biologists, as previously defined. In a first evaluation, each trajectory was considered interesting if any interesting episode is defined in its ground truth. In the context of Machine Learning, it can be seen as a binary classification: it is either classified as interesting or not. Several models, and from different classifiers families, were evaluated to verify which one fits better on this type of data: Support Vector Machine, K-Nearest Neighbors, Naive Bayes, Decision Tree, and Random Forest. Additionally, because of the impact that the pre-processing methods and hyperparameters can have, different

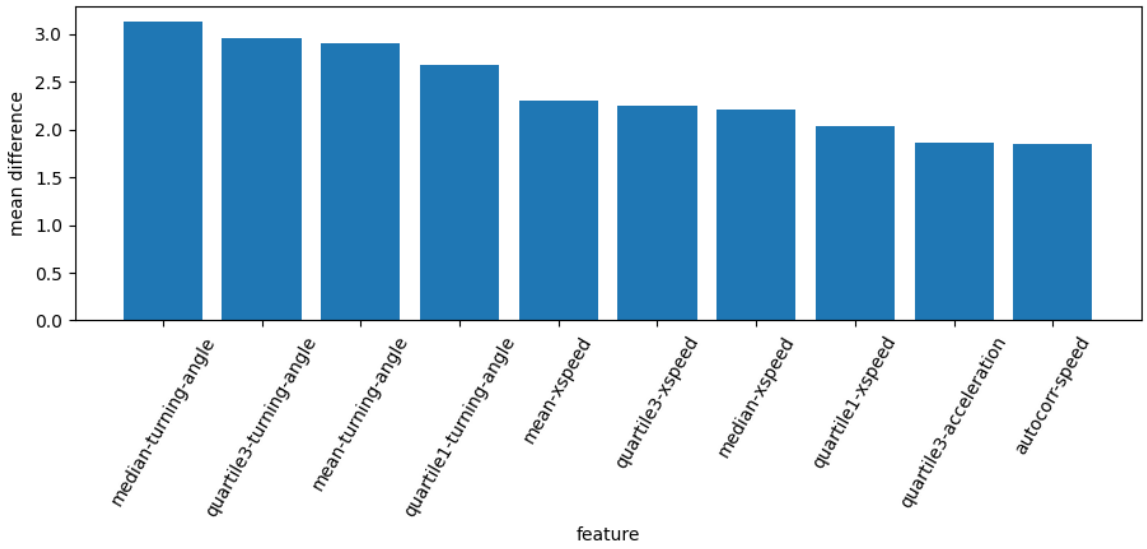

Figure 5.8: Most characterizing features of cluster 0

pipelines were tested, and the respective tuning for each model applying a grid search.

Table 5.2 summarizes the tuning and evaluation process of each of the models. Note that balancing data increased performance in the majority of the cases. It is normal to have more normal trajectories than interesting ones, so generating synthetic interesting samples helped to achieve better results. Also, regarding pre-processing, using the standard scaler normalizer makes the feature weight even which can be important for classifiers that regard similarities and distances. We can also verify that Decision trees and Random Forest have a considerable amount of parameters to tune due to the tree growth settings, that have a high impact on performance. The model that got better accuracy was the Support Vector Machine which achieved 82%.

Model	Best pre-processing	Best parameters	Accuracy
SVM	balancer + normalizer	kernel=poly, c=0.01 degree=3, gamma=1	0.82
KNN	balancer + normalizer + pca	metric=euclidean, n_neighbors=5	0.73
NB	-	-	0.78
DT	balancer	criterion=entropy, min_samples_split=2 min_samples_leaf=1, min_impurity_decrease=0	0.78
RF	-	n_estimators=5, criterion=entropy max_depth=5, max_features=sqrt, min_samples_leaf=3, min_samples_split=2 bootstrap=true	0.77

Table 5.2: Tuning and performance summary of Machine Learning models

Precision and recall can also have an impact on the model choice process. In our context, it would be possibly overwhelming for biologists to analyze several false positives. However, it can also be a danger to let some interesting moments pass by. It is a trade-off that has to be agreed to biologists'

needs. Figure 5.9 draws the precision-recall curve for each of the models. This also complements the information from the previous table and consolidates the fact that the Support Vector Machine works better with this data. Table 5.3 shows the area under the curve values, which illustrate the models that have a better precision-recall overall trade-off.

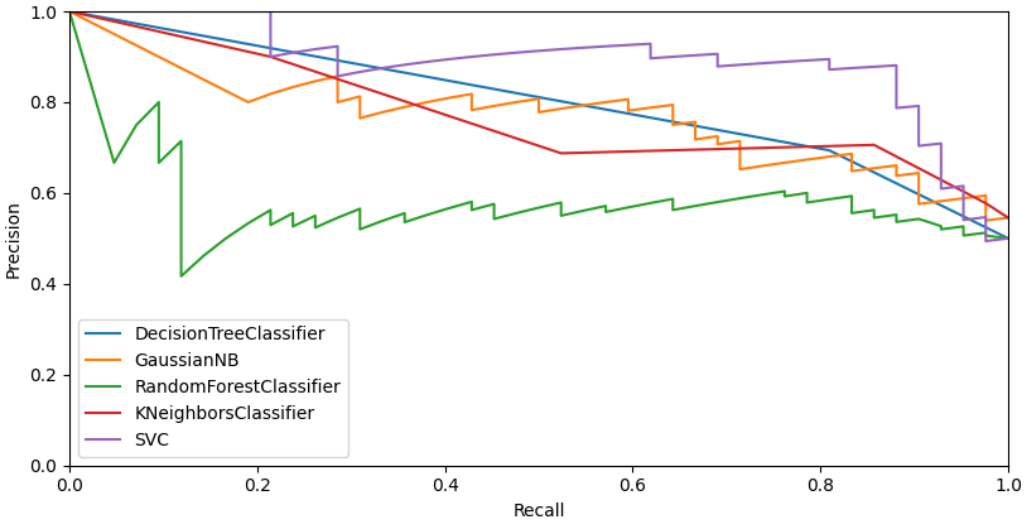

Figure 5.9: Precision-Recall curves for the different Machine Learning models

Model	AUC
SVM	0.87
KNN	0.8
NB	0.8
DT	0.82
RF	0.7

Table 5.3: Area Under the Curve (AUC) values

5.2.2 Model by species

The previous evaluation considers models that are trained using all the samples for both focus species (shark and manta-ray). Figure 5.11 shows the balance of the sample in terms of species. There are a few more shark samples than manta-rays. This experiment regards verifying if separating on different models for each species produces better results. In other words, the objective is to train two different models: one using only the shark samples and another one only using manta-ray samples. The Support Vector Machine was used for comparison purposes, and the best pipeline was verified on the previous evaluation. Figure 5.10 reflects the obtained results. Better performance was achieved modeling using data from both species instead of separating on different models. However, it was not a significant difference and the model for manta rays got a better recall value although the precision has been harmed.

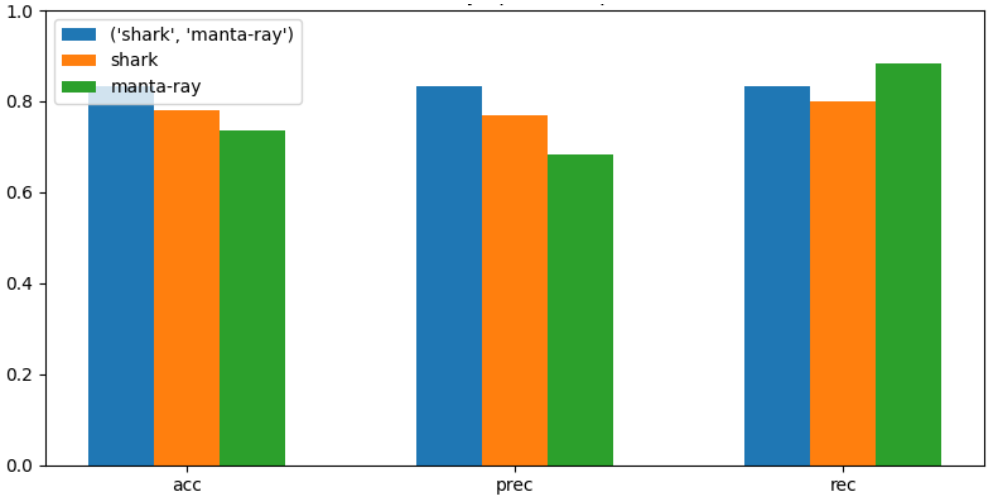

Figure 5.10: Models performance modeling by species

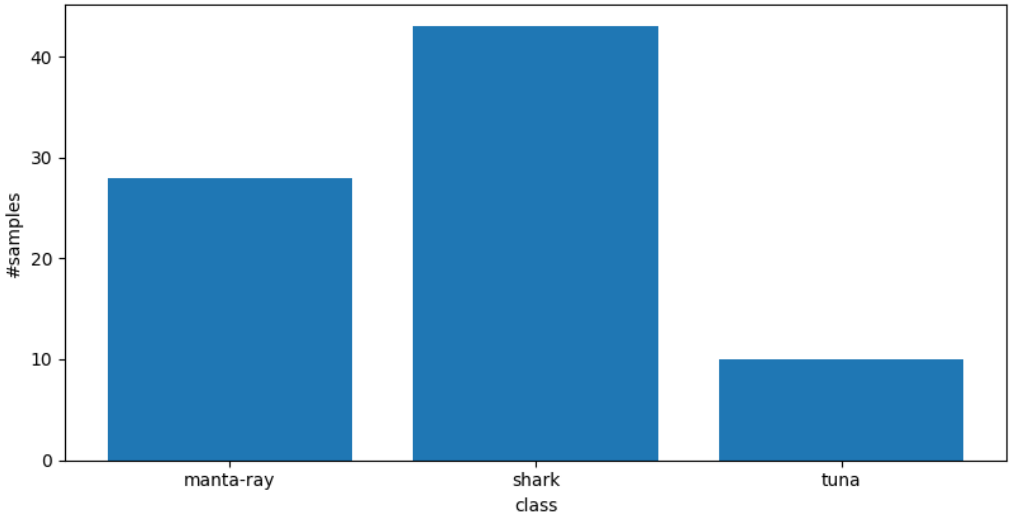

Figure 5.11: Samples by species

5.2.3 Segmentation impact

Segmentation could also be useful for interesting episodes identification. The idea was to segment all the trajectories and to perform classification on each segment instead of the trajectory as a whole. For a given trajectory segment, it was considered to be interesting to be analyzed if there was some interesting episode contained on it. This approach could be a way of avoiding the fact that there are statistics extracted from the features time series that can be attenuated by the values on normal intervals.

The segmentation task followed the Douglass Peucker approach, as explained previously, and values of 30, 2, and 50 were used as epsilon values for position, speed, and angle respectively to avoid over-segmentation. After applying segmentation on all the trajectories set, the class balance changed to the

values illustrated in Figure 5.13. As expected, there were more normal segments but there were also a considerable amount of interesting segments. In terms of the performance difference, similar to the ones observed on modeling by species results were obtained: better performance was achieved without using segmentation but the difference was not significant.

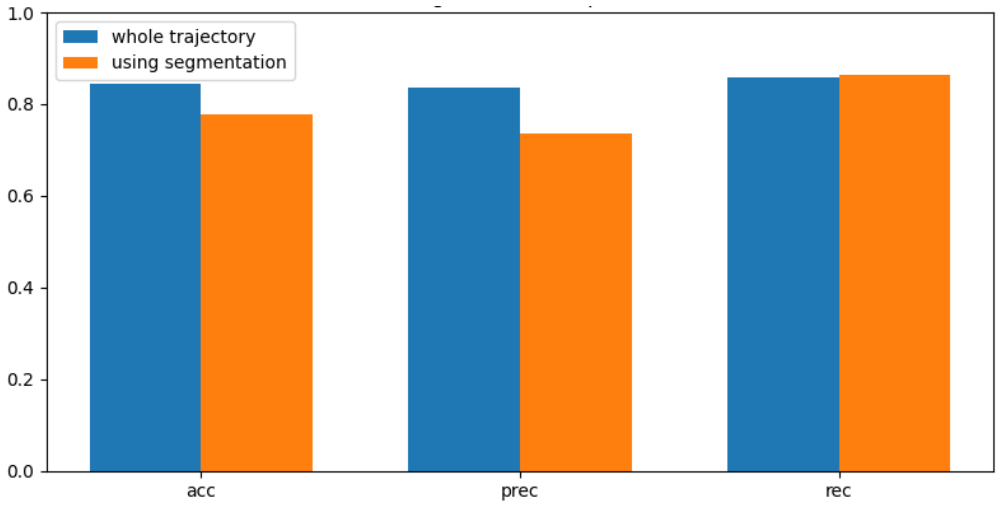

Figure 5.12: Model performance using segmentation

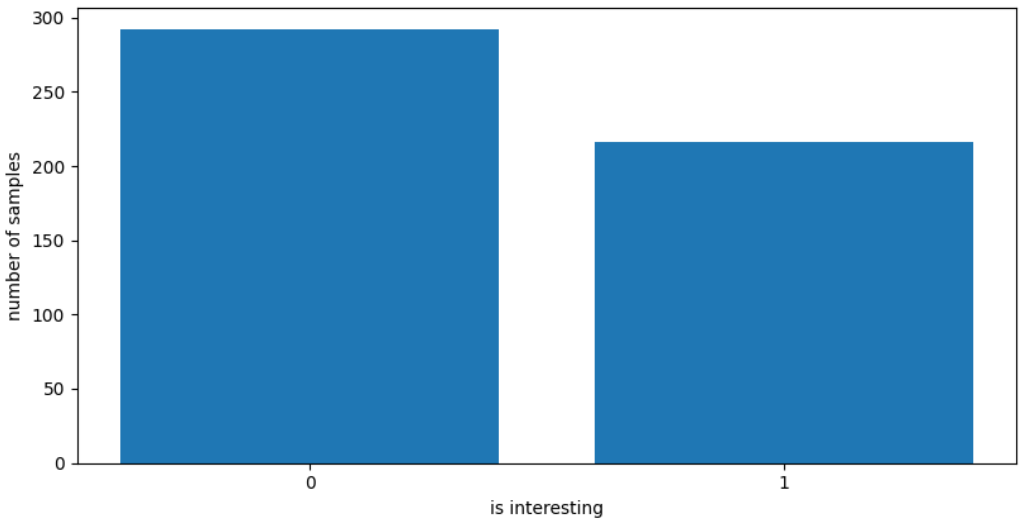

Figure 5.13: Segments class balance

5.2.4 Vector Switching Model

The vector switching model can also be adapted to incorporate this task. The idea is to train two different models which receive as input the two groups of trajectories: normal trajectories and interesting

trajectories. When a new trajectory arrives, the likelihood probability is calculated for both models and the label with the highest probability is assigned.

Before the evaluation itself, a tuning phase was performed to discover the best parameter values for the number of grid nodes, number of fields, delta value, and alpha value. Tables 5.4 and 5.5 show the models that gave better complete log-likelihood. In other words, the models that best fit the trajectories received as input. For both groups, normal and interesting, we could conclude that it worked better with twenty-five nodes (grid of 5x5), four-vector fields, and a delta of one hundred and twenty-five, regardless of the alpha value.

number of nodes	number of fields	delta	alpha	complete log likelihood
25	4	125	2	-7442
25	4	125	1	-7444
16	4	125	1	-7496
16	4	125	2	-7496

Table 5.4: Vector swicthing model tuning results (normal trajectories group)

number of nodes	number of fields	delta	alpha	complete log likelihood
25	4	125	1	-5548
25	4	125	2	-5551
16	4	125	2	-5696
16	4	125	1	-5699

Table 5.5: Vector swicthing model tuning results (interesting trajectories group)

The models that best fit each of the groups were used on the interesting trajectories detection task. Table 5.6 illustrates the number of correctly classified trajectories on each group. As we can see, the results show us a good precision value but a significantly low recall value, which means that this approach gave a considerable amount of false positives, once the overall accuracy stayed by 58%.

group	correctly classified
normal	17 of a total of 42 (40%)
interesting	24 of a total of 29 (83%)
both	41 of a total of 71 (58%)

Table 5.6: Vector swicthing model evaluation results

5.3 Results of experiment C: Feeding Period Detection CNN vs Motion based methods

The feeding period is a group behavior, it depends on the union of the behaviors of all detected fishes in a given moment. As explained in the previous chapters, the focus species, sharks and manta rays,

tend to aggregate when they feel that the feeding period is close. Most of the manta rays are fed on the bottom of the tank through divers and sharks are fed at the surface through sticks that have food on its tip. Using the bottom-feeding videos, both of the approaches, convolutional neural network, and motion-based, were evaluated. Also, evaluation results for the network were gathered using the surface dataset. In this scenario, the motion-based was not experimented with since there is always significant motion on the surface due to the shoals.

5.3.1 Bottom feeding results

Architecture and hyper-parameters can have major importance on classification results. The training samples were split into a training set and validation set, and several models were trained and evaluated. Tables 5.7 and 5.8 show the grid search results (the best seven models out of the sixteen models that were trained). It was only taken into account parameters that were considered to be the most impactful to performance, regarding hyper-parameters and also architecture. It was possible to conclude the following points:

- architecture: best results were achieved using one convolutional layer and two hidden layers or vice versa;
- hyper-parameters: best results were achieved using a learning rate of 0.001 and the mean squared error as error function;
- the model with the best performance (accuracy of 94%) on the validation had the following parameters: number of convolutional layers (1), number of hidden layers (2), number of neurons per hidden layer (120), number of applied filters (5), learning rate (0.001), dropout rate (0%), activation function (relu), error function (mean squared error).

Relatively to the motion methods, both approaches follow the same logic and regard on a threshold definition. Active pixels and average optical flow magnitude are calculated on each instant, and the time series for each of the features was visualized, to decide the most suitable threshold for a given training video that includes normal and feeding frames. On both time series, a well-defined difference can be seen in values regarding these two states. This is illustrated by the example in Figure 5.14. According to this logic, the most suitable threshold for active pixels was defined as 394,000 and for average optical flow magnitude as 2,8.

Using the model parameters that gave the best performance, in the case of the Convolutional Neural Network (CNN), and the predefined thresholds in the case of the motion-based methods, these approaches were evaluated in a different test set. Both approaches gave good results according to accuracy, precision, and recall metrics having been able to reach 92% and 96% of accuracy respectively.

learning rate	dropout rate	activation function	error function	validation set acc
0.01	None	relu	binary_crossentropy	0.56
0.01	0.2	relu	mean_squared_error	0.56
0.01	0.2	relu	binary_crossentropy	0.56
0.001	0.2	sigmoid	mean_squared_error	0.80
0.001	None	sigmoid	mean_squared_error	0.84
0.001	0.2	relu	mean_squared_error	0.85
0.001	None	relu	mean_squared_error	0.90

Table 5.7: Hyper-parameters tuning results for bottom dataset

nº of CL	nº of HL	nº neurons per HL	nº filters	validation set acc
1	2	120	15	0.86
1	2	80	15	0.86
1	2	80	5	0.90
2	1	120	15	0.91
2	1	120	5	0.92
2	1	80	15	0.92
1	2	120	5	0.94

Table 5.8: Architecture tuning results for bottom dataset

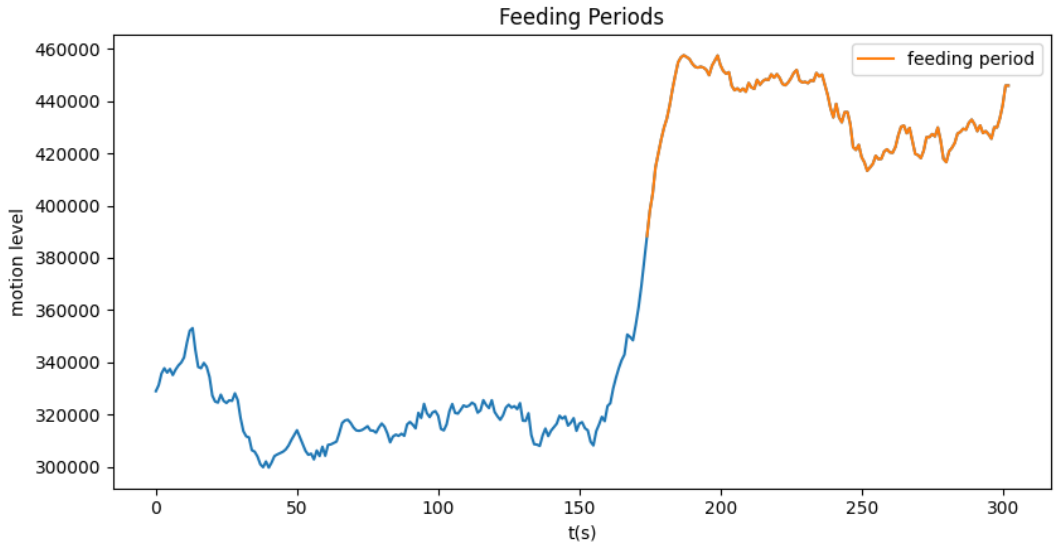

Figure 5.14: Active pixels time series for the training video

However, the optical flow method classified most of the frames as non-feeding frames, which means that the vectors gathered on the test video had a lower average magnitude than the one observed on the training video, even on the feeding period.

Two additional experiences were made: evaluate the motion-based approach using the region definition ability and the CNN in a video filmed with a different camera which was in a different perspective. As we can see, the region was not the problem for the miss classification of some of the frames because

the same results were sensibly observed. Also, we can conclude that the initial trained model to bottom-feeding classification task could not generalize to another perspective. Contrary to what was observed on the optical flow approach, almost all the frames were classified as feeding frames.

As a final analysis, we also tried to understand if the errors that we were getting were close to the feeding state transition. The errors timeline for the CNN approach is drawn in Figure 5.15. As we can see, for this approach, the errors are scattered through time. The same does not happen with the active pixels method. When the feeding period ends, the motion that is detected does not drop instantly, once the frames that are still close to this state change are still classified as feeding frames.

Method	Accuracy	Precision	Recall
CNN	0.92	0.88	0.97
active pixels	0.96	1	0.89
optical flow	0.4	0.05	0.96
active pixels (using region)	0.96	1	0.89
CNN using go pro dataset	0.31	0.998	0.0009

Table 5.9: Bottom feeding test set results

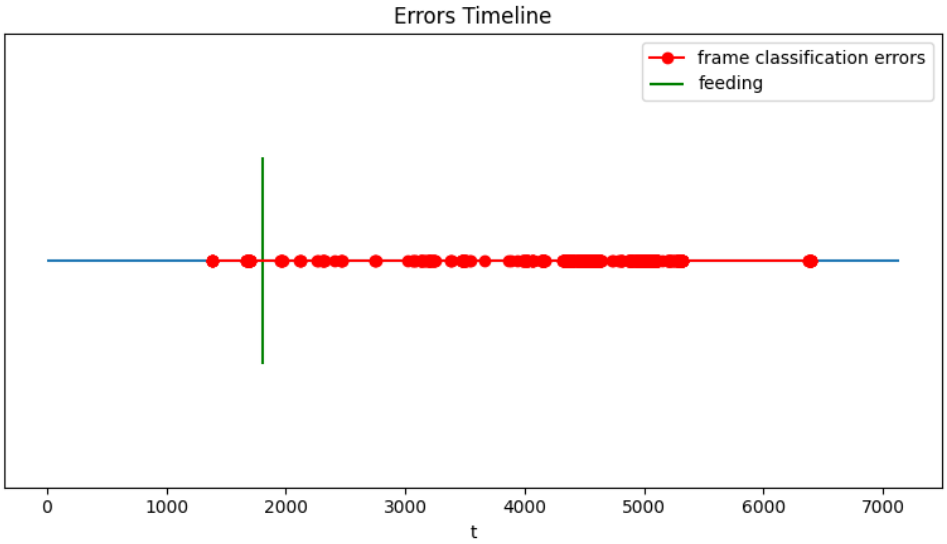

Figure 5.15: Errors timeline using the CNN

5.3.2 Surface feeding results

Surface feeding is more difficult to notice than bottom feeding. When this event happens on the bottom, a clear change in the number of aggregated species is verified, and also on the level of motion in that region. On another hand, surface feeding is hard to visualize. Even as a human, the food sticks are not easily detectable on the frames, only if there is some knowledge about the context. Figure 4.14

illustrates a frame during the feeding period and a frame during the normal period. These example frames have a high resolution which is not the one passed into the network. Even in this resolution, we can conclude that the sticks are hard to verify.

In terms of evaluation, the same methodology was used when comparing to bottom-feeding evaluation. First, we tried to identify the most suitable parameters in terms of hyper-parameters and architecture. After that, using the parameters that gave the best performance on a validation set, the model was evaluated on a different test set.

Tables 5.10 and 5.11 illustrated the tuning results regarding hyper-parameters and architecture respectively. Only the best seven models' parameters are on the table, similar to what was presented in the bottom-feeding results. According to hyper-parameters, there is no significant difference between the presented results. Only three combinations gave a considerable smaller accuracy of 39%:

1. learning rate of 0.001, no dropout, sigmoid as activation function, binary crossentropy as error function

2. learning rate of 0.001, dropout rate of 0.2, sigmoid as activation function, binary crossentropy as error function

3. learning rate of 0.01, dropout rate of 0.2, relu as activation function, binary crossentropy as error function

Looking at the following models, we can see that the binary cross-entropy did not work well with the sigmoid function and a learning rate of 0.001, and also that specific combination using the relu activation function.

More differential results were observed regarding architecture. When using only one hidden layer, the best results seem to be getting together with eighty neurons per layer. On the other hand, using two hidden layers worked better using one hundred and twenty neurons on each. Additionally, as a general rule, using only one hidden layer also gives better results. However, despite all these patterns, 78% of accuracy (best accuracy) was achieved using a combination of two hidden layers with one hundred and twenty neurons each and one convolutional layer, and five filters. Regarding the number of filters, if we change the number of filters to fifteen the accuracy decreases 9%.

The model described previously was also evaluated on a different test set. The obtained confusion matrix is illustrated by Figure 5.16. This model was able to identify correctly all the non-feeding frames giving a maximum recall of 1. Relatively to feeding images, the precision was significantly lower obtaining a value of 0.53. As expected, better results were achieved on the bottom-feeding model.

learning rate	dropout rate	activation function	error function	validation set acc
0.01	None	relu	binary_crossentropy	0.61
0.01	None	sigmoid	mean_squared_error	0.61
0.01	None	sigmoid	binary_crossentropy	0.61
0.01	0.2	relu	mean_squared_error	0.61
0.01	0.2	sigmoid	mean_squared_error	0.61
0.01	0.2	sigmoid	binary_crossentropy	0.61
0.001	0.2	relu	mean_squared_error	0.61

Table 5.10: Hyper-parameters tuning results for surface dataset

nº of CL	nº of HL	nº of neurons per HL	nºfilters	validation set acc
2	2	120	15	0.69
1	2	120	15	0.69
2	1	120	15	0.70
1	1	80	5	0.71
1	1	80	15	0.74
2	1	80	5	0.74
1	2	120	5	0.78

Table 5.11: Architecture tuning results for surface dataset

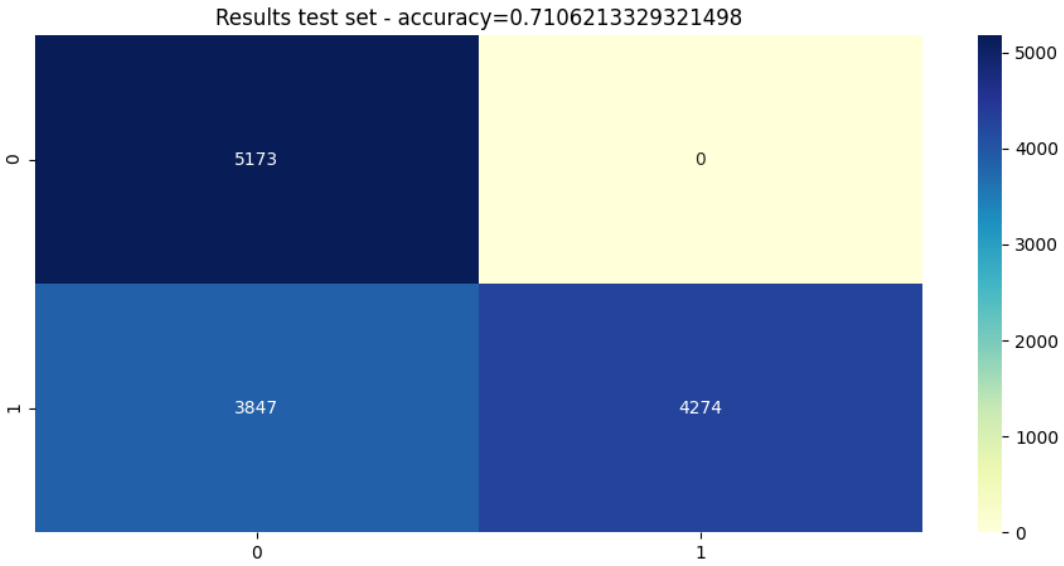

Figure 5.16: Confusion matrix using the test set and for surface dataset

5.4 Experiments Summary

Several approaches were experimented for the various activities. Table 5.12 presents a summary of the evaluation metric value for each of the methods. In green, we have the algorithm that gave the highest performance value for each behavior. Relatively to the anomaly detection, DBScan output more cohesive clusters, and also more separated from each other. Regarding interesting moments detec-

tion, the support vector machine obtained the highest accuracy from the various experimented models. Regarding this activity, we can also notice two points:

- multiple-models and segmentation did not improve the performance;
- switching vector fields gave poor results, when compared with the rest of the models, probably due to the lack of trajectories to train the numerous parameters.

Finally, all the methods achieved good performance on the feeding side, except the utilization of the optical flow. This approach has more noise due to the vector prediction, and the motion threshold is more difficult to adapt to the several videos.

activity	**approach**	**evaluation value (silhouette/accuracy)**
anomaly detection	KMeans	0.30
	DBScan	0.72
interesting moments	K-Nearest Neighbors	0.73
	Naive Bayes	0.78
	Decision Tree	0.78
	Random Forest	0.77
	Support Vector Machine	0.82
	SVM with multiple-model	sharks - 0.78 manta rays - 0.72
	SVM with segmentation	0.79
	Switching Vector Fields	0.58
feeding	Convolutional Neural Network	0.92
	Ative Pixels	0.96
	Active Pixels using a region	0.96
	Optical Flow	0.4

Table 5.12: Experiments evaluation summary

6

Conclusion

Aquatic life has a lot of density. It can be hard for biologists to track all the fish activity. Automatic behavior detection can play a major role, which can save time for biologists, and let them focus on other tasks. Traditionally, it is made by visual inspection, and that requires the biologists to spend considerable time analyzing fishes, and it can be subjective to biologist experience. In this project, we focused on abnormal behaviors, feeding periods, and interesting moments of sharks and manta rays, due to their importance to Lisbon Oceanarium. Keep track of abnormal behaviors can be mainly important to conclude about diseases, water quality problems, or poor habitat integration. Additionally, feeding can also impact water quality, and it is important to control food waste that can lead to costs or underfeeding that can lead to aggressive behaviors.

In this project, it was developed a system capable of detecting these behaviors in the main tank of the oceanarium, and as described, focusing on sharks and manta rays. To detect abnormal and interesting moments we extract several features from each fish trajectory and we try to model this kind of behavior through machine learning models, and a switching vector model. Additionally, the system also has the ability to define a set of rules based on those features. Relatively to feeding, there are several factors that can describe this behavior: we can define a threshold according to aggregation or motion variability, or try to model feeding frames patterns through a convolutional neural network.

Overall the system achieved good performance metrics. The detected outlier seems to also correspond to an interesting moment, given its sudden changes in direction, but it would be good to have a more robust set of trajectories. In terms of its evaluation, a clustering algorithm based on density is considered more appropriate for this task. Regarding interesting moments, all the training models round the 80% of accuracy. The usage of segmentation or multiple-model training did not make any significant difference in terms of achieved results, and the switching vector model adaption obtained poor results regarding this task. Finally, for the feeding behavior, convolutional neural networks could model feeding image patterns and achieved 92% of accuracy. The motion approach, based on active pixels identification, also obtained results in the excellent range (96%). On the other hand, using optical flow to this effect was noisier decreasing this value to only 40%.

There are several topics that could be explored in the future. One of them could be focusing on the clustering approach, but using a fish trajectories big data dataset. Additionally, all the methods suffer from image plane dependency and camera perspective, especially for the trajectories definition which is a sequence of 2D position points. Another future problem could be focusing on fixing or decreasing this dependency. Results coming from the convolutional neural network approach can be hard to understand, it would be also interesting to try to understand what image patterns, and image regions, are characterizing each of the classes. Finally, but less interesting in an intelligence computation area, is focusing on biologists' contact. It would be extremely useful to pick up all the developed projects and modules and develop an application to be used by biologists.

Bibliography

[1] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis *et al.*, "Semantic trajectories modeling and analysis," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, pp. 1–32, 2013.

[2] C. Barata, J. C. Nascimento, and J. S. Marques, "Improving a switched vector field model for pedestrian motion analysis," in *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, 2018, pp. 3–13.

[3] T. J. Pires and M. A. Figueiredo, "Shape-based trajectory clustering." in *ICPRAM*, 2017, pp. 71–81.

[4] J. Castelo, H. S. Pinto, A. Bernardino, and N. Baylina, "Video based live tracking of fishes in tanks," in *International Conference on Image Analysis and Recognition*. Springer, 2020, pp. 161–173.

[5] J. Santos, H. S. Pinto, and A. Bernardino, "Tracking animals in underwater videos," Ph.D. dissertation, Instituto Superior Técnico, Lisbon, 2020.

[6] V. M. Papadakis, I. E. Papadakis, F. Lamprianidou, A. Glaropoulos, and M. Kentouri, "A computer-vision system and methodology for the analysis of fish behavior," *Aquacultural engineering*, vol. 46, pp. 53–59, 2012.

[7] J. Zhao, Z. Gu, M. Shi, H. Lu, J. Li, M. Shen, Z. Ye, and S. Zhu, "Spatial behavioral characteristics and statistics-based kinetic energy modeling in special behaviors detection of a shoal of fish in a recirculating aquaculture system," *Computers and Electronics in Agriculture*, vol. 127, pp. 271–280, 2016.

[8] C. Zhou, B. Zhang, K. Lin, D. Xu, C. Chen, X. Yang, and C. Sun, "Near-infrared imaging to quantify the feeding behavior of fish in aquaculture," *Computers and Electronics in Agriculture*, vol. 135, pp. 233–241, 2017.

[9] C. Zhou, K. Lin, D. Xu, L. Chen, Q. Guo, C. Sun, and X. Yang, "Near infrared computer vision and neuro-fuzzy model-based feeding decision system for fish in aquaculture," *Computers and Electronics in Agriculture*, vol. 146, pp. 114–124, 2018.

[10] C. Spampinato, D. Giordano, R. Di Salvo, Y.-H. J. Chen-Burger, R. B. Fisher, and G. Nadarajan, "Automatic fish classification for underwater species behavior understanding," in *Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams*, 2010, pp. 45–50.

[11] O. Anas, Y. Wageeh, H. E.-D. Mohamed, A. Fadl, N. ElMasry, A. Nabil, and A. Atia, "Detecting abnormal fish behavior using motion trajectories in ubiquitous environments," *Procedia Computer Science*, vol. 175, pp. 141–148, 2020.

[12] T. G. D. Coleman, "Sharkid: A framework for automated individual shark identification," Ph.D. dissertation, California State University, Long Beach, 2020.

[13] S. Ogunlana, O. Olabode, S. Oluwadare, and G. Iwasokun, "Fish classification using support vector machine," *African Journal of Computing & ICT*, vol. 8, no. 2, pp. 75–82, 2015.

[14] D. Rathi, S. Jain, and S. Indu, "Underwater fish species classification using convolutional neural network and deep learning," in *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. IEEE, 2017, pp. 1–6.

[15] C. Yu, X. Fan, Z. Hu, X. Xia, Y. Zhao, R. Li, and Y. Bai, "Segmentation and measurement scheme for fish morphological features based on mask r-cnn," *Information Processing in Agriculture*, 2020.

[16] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[17] C. Beyan and R. B. Fisher, "A filtering mechanism for normal fish trajectories," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 2286–2289.

[18] ——, "Detection of abnormal fish trajectories using a clustering based hierarchical classifier." in *BMVC*, 2013.

[19] F. Broell, T. Noda, S. Wright, P. Domenici, J. F. Steffensen, J.-P. Auclair, and C. T. Taggart, "Accelerometer tags: detecting and identifying activities in fish and the effect of sampling frequency," *Journal of Experimental Biology*, vol. 216, no. 7, pp. 1255–1264, 2013.

[20] W. Zhang, A. Martinez, E. N. Meese, C. G. Lowe, Y. Yang, and H.-G. Yeh, "Deep convolutional neural networks for shark behavior analysis," in *2019 IEEE Green Energy and Smart Systems Conference (IGESSC)*. IEEE, 2019, pp. 1–6.

[21] C. Zhou, D. Xu, L. Chen, S. Zhang, C. Sun, X. Yang, and Y. Wang, "Evaluation of fish feeding intensity in aquaculture using a convolutional neural network and machine vision," *Aquaculture*, vol. 507, pp. 457–465, 2019.