Tracking Fishes in Tanks (or can Fishes become musicians?)

José Castelo
IST, 80866
josecastelo@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal
https://tecnico.ulisboa.pt/

Abstract. Detecting, tracking, and classifying fishes in tanks are processes involving multiple domains such as image processing, computer vision, and artificial intelligence. If done successfully, they can be the foundation for other systems: fish health monitoring, interactive entertainment/informative systems and others. In this document related approaches and techniques are reviewed and compared. Problems associated with these approaches are identified and analyzed. Then a new approach for tracking and classifying fish swimming in aquariums in real-time is proposed. An evaluation method for this proposal is defined and a schedule regarding the work of this thesis is outlined.

Keywords: Image Processing · Object Detection · Object Tracking · Object Classification · Fish.

Table of Contents

Tracking Fishes in Tanks (or can Fishes become musicians?) 1

1 Introduction 1

2 Related Work 2

  2.1 Review 2

  2.2 Analysis 8

3 Proposal 11

  3.1 Conditions and Requirements 11

  3.2 System Pipeline 12

  3.3 Detection Module 14

  3.4 Tracking Module 15

  3.5 Classification Module 17

4 Evaluation Method 18

5 Schedule 18

6 Conclusions 18

1 Introduction

Tracking fish in videos has applications ranging from monitoring their behaviour to entertainment purposes. However, it is not a simple process, involving multiple domains in the realm of computer vision, image processing and artificial intelligence. Our goal is to provide reliable tracking and classification of fish swimming in aquariums in real-time. This process can be divided into three major sub-processes: detection, tracking and classification. Object detection, tracking and classification in videos have been areas of increased interest due to their valuable applications in real-world scenarios such as surveillance and automated vehicle systems. Each of these areas has its own goal, as well as different problems to overcome. However it seems that as these techniques are result-driven they tend to become more and more computationally expensive, turning some of them unfeasible to run in real-time applications.

The detection sub-process has the goal of detecting instances of fish present in images (each considered frame of a video stream, in our case). Object detection is achieved by identifying which regions of an image make up the background, and which belong to objects of interest. In the case of videos (sequences of images) this is usually done in two ways: based in a description of the background (or the objects), or based on movement. Each method has to be considered depending on the properties of the video. Properties such as static background, objects movement (or lack of it) should be considered, as they influence whether each type of method is applicable.

The process of tracking objects in videos seeks to provide the location of objects as time passes. Usually the tracking is guided by the detection of objects but it can not rely exclusively on that information. When tracking multiple objects each detected object has to be matched to its track, which has multiple issues. Problems such as objects overlapping, going out of frame, being partially or totally occluded (hidden behind something), and unpredictable movement all contribute to the complexity of tracking objects.

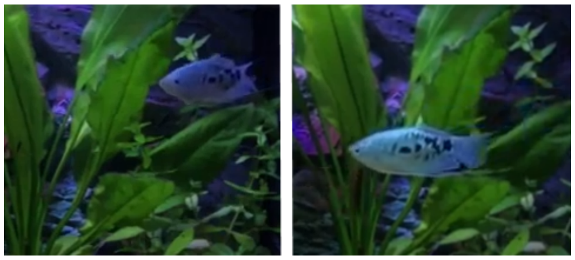

Fig. 1: An example of lighting affecting the appearance of a fish.

The process of classifying objects has the goal of identifying which class of object an object belongs too. In our case the goal is to identify which species a

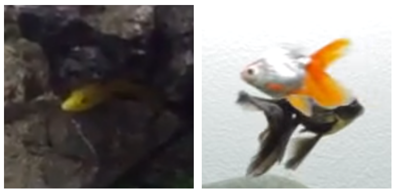

Fig. 2: An example of a fish partially occluded behind a rock and two fish overlapping.

fish belongs to, knowing which species are present in the environment. In order to achieve this, features of the tracked fish must be compared and matched to the features of each species. Problems such as intra-class classification (classes of objects that share the same super-class, e.g. multiple species of fish) which have similar features, and real-time constraints make this process challenging.

In section 2 a review and analysis of research work with similar goals is presented to allow an understanding of the state of detection, tracking and classification techniques. Then, based on these techniques a proposal is made in section 3 with adequate methods according to the conditions, restrictions and goals of our particular setting. An evaluation method for our proposal is suggested in 4.

2 Related Work

In this section research work related to the theme of this thesis, in the areas of object detection, tracking, and classification is reviewed and then analyzed.

2.1 Review

Recently the authors of [13] compared the performance of two supervised machine learning approaches to detect and recognize coral fishes in underwater videos, with the main goal of surveying different species population pools in a given region.

Much like other underwater video scenarios, problems such as color and lighting variations as well as sediments or floating debris (implying a moving background) had to be taken into consideration.

One of the tested approaches was the use of a Histogram of Oriented Gradients [3] (HOG) with a Support Vector Machine [12] (SVM). A HOG describes an object by computing the histogram of directional changes in intensity (oriented gradients) of a given image. As such, it contains information derived from the edges of objects causing sharp changes in pixel intensity. A SVM is a supervised method able to classify feature vectors (such as a HOG) by separating each class of interest, as well as possible, in a high dimensional space.

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

Instead of using a simple network such as described above the authors decided to use a more complex one called Convolutional Neural Network [7] (CNN) which allows the network to extract the best features on its own. It works by using convolutional layers, which, by using a convolutional kernel morphs the signal. Then it pools the signals (usually by region) which reduces the dimensionality problem and improves performance by filtering the most relevant bits of a region (such as the maximum value, for example). The last convolutional layer will eventually output a value, representing the probability of the image belonging to each class, based on the summary of the most relevant information already gathered by the previous chain of layers.

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

Another study with a similar goal is described in [10]. The authors sought to automate fish population surveying through the analysis of videos. However, in this case the authors focused on an unconstrained video environment.

In order to detect fish a moving average (MA) background subtraction algorithm was applied. Given a background reference in time-step t, B_t, and a frame, F_t, the MA flags pixels as belonging to the foreground (or in other words, belonging to a moving object of interest) if the absolute difference in value of a given pixel between images F_t and B_t is greater than a given threshold T_a.

$$|B_t(x,y) - F_t(x,y)| > T_a$$

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

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

However, this is not enough to stop detecting repetitive movement from moving objects. In order to overcome this problem the authors filtered the false positives through the use of Adaptive Gaussian Mixture Models. The choice was a balance between not having to carefully calibrate any parameters and being computationally lightweight enough. An Adaptive Gaussian Mixture Model [4] (AGMM) is able to describe multimodal variables such as the intensity value of a pixel that is seen when a repetitive background motion is present (e.g.: an aquatic plant waving in the water).

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

$$SelectedPix(frame) = MovingAverage(frame) \cap FitAGMM(frame)$$

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

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

The work developed in [6] regards tracking and classifying vehicles and pedestrians through videos of an open environment with a static camera. The pipeline proposed by the authors is composed of three distinct modules: motion segmentation, where moving objects are detected; object tracking, where each individual object is tracked as time goes; object classification, where each object is classified as being a person, a vehicle, or other.

In the motion segmentation module foreground pixels are first identified by comparing the difference in values of the frame in question with the background reference image, according to a threshold (determined experimentally), similarly to the work in [10], previously reviewed. The background reference image is built and updated with an Approximated Median Filter [9] (AMF). This filter computes the approximated median value within a neighborhood of a pixel and nudges the pixel value of the background reference towards that value. Authors mention that adding a step factor to that change according to the difference magnitude between these two values, such that the value adjustment is higher for bigger discrepancies, improves the results. Once the foreground pixels are identified they are segmented into separate objects by using a two-pass connected component labeling method, which identifies clusters of foreground pixels belonging to the same object of interest. Finally bounding boxes for each object are built based on the clusters obtained from the connected components.

The object tracking module of this project is split into four steps. First an object color model is built by computing a normalized RGB color histogram (count of each color value within the bounding box of each object). Then the position of the object for each previously existing track is predicted by a Kalman Filter [5], which, based on the position history at each time-step attempts to account for errors and provides a new estimate position. The velocity of each object is assumed to be nearly constant but noise is accounted for. The third step is an A-Priori Assignment of each object to a track. For each existing track (from previous frames) and detected object (from the segmentation module) a distance matrix is built by computing the distance from each predicted position (based on the track, from the previous step) and the actual position of each detected object in the frame. By checking which cells contain the minimum value in each row and column we are able to pair objects with existing tracks, storing their correspondence value (the Bhattacharya distance [1], which is a correlation value between the RGB histograms of the tracked object and the detected object). Objects are assigned to existing tracks if their correspondence value is above a certain threshold. A last step merges tracks that are too close (overlapping objects), splits tracks that were merged and whose objects have now diverged in position, assigns new tracks to unmatched objects (when a new objects enters the scene) and tracks that remained unmatched for too long are forgotten (when an object leaves the scene).

In the object classification module the authors have opted to classify each object based on its repetitive motion (defined as repetitive changes in shape) since walking pedestrians show this kind of motion whereas a vehicle is mostly static in shape. For each new frame at time-step t, and after accounting for changes in bounding box sizes (scaling operations are applied) a new Difference Image (D_t) is computed for each tracked object, i, as an exclusive-or operation between the values of pixels of the previous bounding box (O_{t-1}) and the current one (O_t).

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

This means that a D will contain ones in pixels which have changed in value (accounting for noise with a filter threshold) and zeros in pixels which have not. A Recurrent Motion Image is the sum of all D for a given object, in a given time interval with n frames. It has higher values in regions that change the most.

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

Finally, a Motion History Image [2] (MHI) is computed by giving regions (block of pixels) that have an average value higher than a threshold (computed experimentally) a value of one and the rest a value of zero. A Motion History

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

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

The work in [11] proposes a real-time adaptation process for the Adaptive Gaussian Mixture Models reviewed before. With the use of approximations, the authors are able to more suitably model the background of complex scenes while keeping the computational complexity low enough to allow use of this technique online.

Instead of trying to compute the Gaussians that strictly model values pertaining to background colors for a given region the authors suggest computing Gaussians that model the most common values and then decide which are most likely to correspond to background colors, based on the persistence and variance of each Gaussian in a given mixture. The need to use multiple Adaptive Gaussians (a mixture) arises when more than one surface (background or not) can be seen in a particular region over time. Given a set S of k Gaussians that each models ω_i, an estimated portion of the color values seen in a given pixel during n consecutive frames of a scene, the authors compute the probability of observing a value x, observed in a new frame at time-step t as:

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

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

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

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

where $M_{k,t}$ is one if the current pixel value is matched with that distribution or zero otherwise. A match occurs when the value is within $T_m = 2.5$ standard deviations from the mean of a distribution. This means that the model adjusts exponentially to changes in distributions and that sporadic erroneous changes are also corrected faster. To decide which distributions describe the background they are first ordered by their value of ω/α, which is higher as the distribution describes a higher amount of values and has less variance. Then the first

J distributions are chosen as background distributions (the authors name this technique Background Model Estimation) such that they describe T_r , the minimum portion of data to be taken into account as background:

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

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

The authors in [8] exploit the synergy between Temporal Differencing of frames and Image Template Matching, with the goal of tracking and classifying humans and vehicles in streams of unstructured outdoors scenes, in a similar fashion to the work done in [10]. The main difference between these two approaches lies in how the tracking and classification is done. In order to more robustly track each object, the authors first classify each blob (note that in [10] the authors classify last). The idea is that it is easier to match each track to the corresponding object if the information regarding which class it belongs to is available.

Classification is done based on the dispersedness of each blob. Dispersedness can be seen as the complexity of the shape in relation to its area (humans being more complex in shape than vehicles).

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

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

Instead of classifying based on a single observation (at each frame, independently) an histogram of the classification for the last n frames is kept and the peak is the class assigned (a simple form of maximum likelihood estimation). This helps when an object is briefly misclassified since the history in n compensates for it.

Once every blob has been classified, they are matched with the existing tracks by means of image template matching (correlation between image templates). Each track has an associated image template reference (R) which is updated with an Infinite Impulse Response filter at each time-step t, according to the current blob image M which is obtained from the detection step.

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

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

2.2 Analysis

First, it is important to point out that only two of the reviewed related work, [13] and [10], are directly comparable with the work to be done in this thesis. Other approaches reviewed in the last section share the abstract problem, whether in the domain of object detection and/or tracking or classification. Furthermore, the major constraint of having to run in real time means that some of these approaches cannot be applied for the problem at hand. Regardless of our specific constraints, there are some ideas that we can borrow from each of them.

Detection The most common technique to use for object detection is some form of Background Subtraction, as seen in section 2.1, since it allows focusing the tracking/classification methods in regions of the frame that are more likely to be on target (belonging to an object of interest), instead of having to run them on the whole of each frame. Which not only introduces potential false-positives (due to background regions with similar features to those of the target objects) but also requires more processing power.

Background Subtraction consists in having a model of the background and then, by comparing it to the new image, selecting regions that differ from it. This model can be either a statistical description (in [10],[6] and [11]) or an image reference itself (in [10], [8]). In the case of a statistical description the comparison checks whether the values of the new image fit the description or not. This approach is robust since it allows distinct intervals of values to be considered (applied through the use of Adaptive Gaussian Mixture Models, for example) entailing that problems such as the detection of background repetitive motions are avoided. However, objects that move slowly represent a problem since they pollute the model with noise. While this is not usually a problem for the simpler Background Reference Image approach, misclassifying regions where an object has just been identified as foreground is, due to changes in values within those regions. It works by directly comparing the new image with the background reference and checking if the values are similar, such as in [8] and [10]. This is the reason why [10] uses both types of approaches at the same time. When combined together, they offset each others shortcomings and produce a better output than any of them alone.

In other approaches, as seen in [13], detection takes place by dividing each frame in multiple regions of variable resolution and checking how likely an object is to be present by comparing its features against the ones found by either CNN or SVM that most accurately describes the training dataset. However this means that a larger area has to go through the more demanding algorithms which means that despite the usual better results, this approach is rarely applicable for real-time applications. Another common issue is that neighboring regions could be detecting only parts of an object, or, two similar objects might be in adjacent regions and thus the merging and splitting of regions has to be taken into consideration, making it harder to get accurate bounding boxes for objects.

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

In [10] an object observation is assigned to an existing track if the value of feature similarity functions (Blob Shape Features or Color Histogram) are above a predetermined threshold. While some accuracy might be lost because no position is predicted it should still perform well in situations where the objects are not moving too fast as to cross paths in a single time-step, and possibly perform even better in situations where the movement is unpredictable, as in our case.

In [8] each track has an associated image template that is updated as new observations for the associated object are made. An observation is associated to a track if the correlation between the image template and the observation is high enough. This approach seems adequate when objects are distinct in appearance. However it is much more prone to drifting if the detection step is not performing well enough.

The method used in [11], a linear predictive multiple hypothesis tracking, assumes that the movement of each object is linear (constant velocity, no acceleration). This means that while results are much better when the assumption is true it can perform worse than other simpler methods when it is not.

A variation of a multiple hypothesis tracking (named A Priori Assignment by the authors) used in [6] makes the previous assumption too. However, it also takes into account a color description of the object making it more robust. If a more suitable prediction method is used it could be a good approach for my case. The authors even implement the merging and splitting of tracks to handle objects overlapping, which is one of the biggest issues that seem to occur in fish tracking (as they regularly swim along/past each other).

Classification From all reviewed research, the only one actually using formal classification algorithms is [13]. All others mostly leverage the performance of their detection and tracking. This allows the classification to be simple yet effective since the regions to be classified are known to be objects. And thus, only requiring to be distinguished among themselves, which seems more inline with the constraint of real-time support.

In [13], a support vector machine approach is used to classify the features extracted by an HOG of a given region. While the authors concluded that the other approach (analyzed next) provides better results, they did not seem to take into consideration that an SVM (or most classification algorithms, for that matter) are only as good as the features/metrics used. The fact that only an HOG was used seems inadequate when there are other features that can be easily extracted such as the color histogram (seen in multiple other similar works, with good results). However, since the SVM was used as a baseline I can see why they would not want to spend too long looking for better features/metrics.

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

The classification seen in [11] is an extreme example of relying on good motion segmentation/tracking methods and taking advantage of the temporal aspect. The authors simply classify each object based on the average ratio between width/height over a time interval, achieving good results. This means that, while some accuracy might be lost, the real time performance becomes easily achievable while leaving more room to improve detection and tracking.

In [6] authors take advantage of the fact that the target object classes (pedestrians and vehicles) change their shapes as they move (pedestrians constantly change their shape by moving their legs while vehicles remain relatively constant in shape). Once again this is only possible when taking the temporal aspect into account (as a single observation would mean no access to this information). The classifier is also a simple binary threshold of the Recurrent Motion Ratio.

The method used in [8] is similar to the previous one. The authors use the complexity of an objects appearance perimeter (dispersedness). A multiple hypothesis approach is used to take into account the temporal aspect. Objects that are assumed to be the same over consecutive frames (based on proximity) are only classified as a certain class if the dispersedness remains in the bin of the same class in both instances. This is another example of exploiting the temporal aspect while keeping the classification metric simple yet effective.

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

Summary Tables 1, 2 and 3 summarize the reviewed techniques in three parameters: real-time performance, which is how suitable the technique is to real-time performance; usefulness for our problem, which is how well the technique suits our specific problem; ease of configuration, which is how easy the technique is to implement and configure (this includes any necessary model training and/or parameter adjustments, when applicable).

Technique	Real-time Performance	Usefulness for our problem	Ease of Configuration
AGMM	++	+++	++
TD	+++	++	+++
CNN	+	++	+
HOG + SVM	++	++	++

Table 1: Comparison between detection techniques: AGMM (Adaptive Gaussian mixture models), TD (Temporal difference), CNN (Convolutional neural network) and HOG + SVM (Histogram of gradients, support vector machine).

Technique	Real-time Performance	Usefulness for our problem	Ease of Configuration
BSF + CH	+++	++	+++
IT	++	+++	++
LPMHT	++	++	+
AA	++	+++	++

Table 2: Comparison between tracking techniques: BSF + CH (Blob shape and color histogram feature matching), IT (Image template correlation), LPMHT (Linear predictive multiple hypothesis tracking) and AA (Apriori assignment).

3 Proposal

3.1 Conditions and Requirements

First, it is important to clearly identify the environment for this work, its system restrictions as well as environment conditions associated with it. The ultimate goal is to create a system that is able to reliably track and identify fish swimming in an enclosed aquarium.

Technique	Real-time Performance	Usefulness for our problem	Ease of Configuration
HOG + SVM	++	++	++
CNN	++	+	+
BB W/H	+++	++	++
RMR	+++	++	++
Disp.	+++	++	+++

Table 3: Comparison between classification techniques: HOG + SVM (Histogram of gradients, support vector machine), CNN (Convolutional neural network), BB W/H (Bounding box width/height ratio), RMR (Reccurent motion ratio) and Dispersedness.

Environment Conditions

1. The video input is captured with a camera that is static in both position and orientation.
2. The video input is in color.
3. Lighting within the aquarium can be variable (in both hue and intensity).
4. A training data set containing no less than 5 example images per fish species present in the aquarium has to be provided.

System Requirements

1. Each fish species in the aquarium should be distinguished by the system. If the classification proves to be too unreliable, then species should be grouped based on features (such as size or species family).
2. The system must run in real-time, providing the position of each tracked fish with a frequency $\geq 5\,\text{Hz}$.
3. The system should support resolutions up to 1280x720 pix.

3.2 System Pipeline

As seen in figure 3 the proposed system will be composed of three modules with distinct goals. The detection module will identify objects of interest in each frame and output their bounding boxes. The tracking module will match objects of interest found by the detection module with the existing tracks, as well as creating new tracks for newly detected objects and deleting no longer relevant tracks. The classification module will finally identify which class (species of fish) each object belongs to. Both tracking and classification modules will need the blob data (bounding box and corresponding values), as well as the history for each track (temporally ordered sequence of blob data) which is represented by the track data portion of the figure.

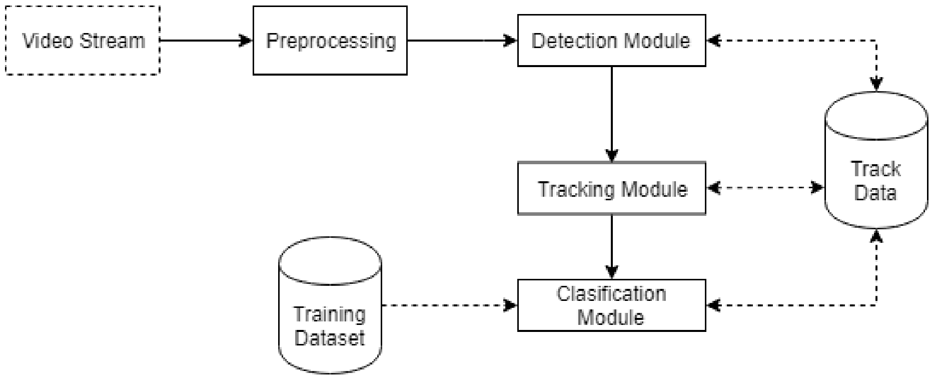

Fig. 3: System pipeline overview

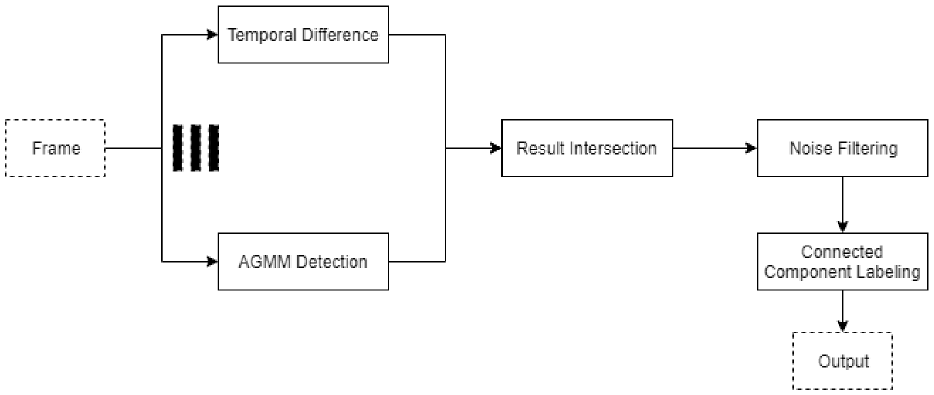

Fig. 4: Detection module overview

3.3 Detection Module

Following the tested and proven approach of [10] it was decided to use both forms of background subtraction: a statistical description method (AGMM) and a temporal difference method (TD), followed by the intersection of both results.

Assuming that $I_t(x,y)$ are the pixel intensity values for each color in a frame at time step t in position (x,y), the temporal difference output TD is simply:

$$TD_t(x,y) = \begin{cases} 1, \text{if } |I_t(x,y) - I_{t-1}(x,y)| > T_{td} \text{ for at least one color channel} \\ 0, \text{otherwise} \end{cases}$$

where T_{td} should be 15% of the intensity value range (according to [8]), although subject to change.

The second form of background subtraction, through the use of Adaptive Gaussian Mixer Models, requires two steps:

First, a model of the background must be built and kept updated: for each pixel, (x,y), a set of up to K Gaussian distributions is kept in order to model its intensity values. At each frame a matching vector $M_t(x,y)$ is built. An intensity value $I_t(x,y)$ is said to match a Gaussian distribution k if it is within T_g standard deviations of its mean value, μ_k. A matched distribution will have an M_t value of 1, others a value of 0. If no match occurs, the distribution with lowest weight is forgotten and a new distribution with high variance, low starting weight and median value $I_t(x,y)$ is made. A vector ω_t describes the weight of each distribution and is updated according to the matching vector M_t and learning rate α as follows:

$$\omega_t = (1-\alpha)\omega_{t-1} + \alpha * M_t$$

ω_t is then re-normalized. The second step of modeling the background is to estimate which of the distributions describe it. A distribution with low variance and a high weight is more likely to be related to the background and background colors should be relatively constant, in contrast to the colors associated to moving objects which are transient. Thus the distributions are ordered by highest values of $\frac{\omega}{\sigma}$ and the first N distributions are chosen such that their commulative weight exceeds threshold T_{mw}:

$$N = argmin_n(\sum_{k=1}^{n} \omega_k > T_{mw})$$

T_{mw} can be seen as the minimum ratio of (recent, controlled α) past values the background distributions must model. As stated by the authors of [11], and seen in the related work, some assumptions are made in order to reduce the complexity of needed computations. This completes the background modelling step. A pixel is then said to belong to the background if it matches any of the updated background distributions.

The TD output is then filtered by the output of the AGMM. At this point the pixels where foreground is present have been flagged. In order to reduce noise (false-positives) even further, flagged pixels that are isolated are unflagged. The method to be used is still not set at this point.

The last procedure in the detection module is to identify which pixels form individual blobs. A connected component labeling algorithm will be applied to achieve this. In its simplest form each flagged pixel from the de-noised output matrix is uniquely labeled and that label is spread to neighboring flagged pixels recursively. The process is repeated until there are no flagged pixels left unlabeled. Then, pixels with the same label are said to form a blob.

The final output of the detection module should be a set of blob descriptions. A blob description is composed of the corresponding rectangular bounding box (position, width and height), a matrix with the RGB values for the bounding box and a matrix with the mask shape (ones for pixels belonging to the object and zeroes otherwise). This data amounts to the "detected blobs" portion of the track data.

3.4 Tracking Module

Given the output from the detection module, a set of blob data, tracking will be achieved through feature matching. The biggest issue faced when attempting to track multiple objects, as discussed before, is handling object occlusion and overlapping.

The features to be considered for tracking fish are:

1. The centroid, C, of each blob, corresponding to its position.
2. The area, A, of each blob. It is equal to the number of ones in the mask of the blob.
3. The binned RGB histogram, H, is a simple and relatively computationally inexpensive image description feature.

These features will be used to decide how similar a blob is to a tracked fish. A track T consists of an ordered set of blobs (one for each consecutive considered frame, at most). Thus it can be directly compared to a blob if only considering one of its own blobs. If we take the most recent blob belonging to a track, and its features are similar to a detected blob, it is likely that they regard the same fish. However, to do that, a similarity function for each feature is necessary.

In order to compute a similarity value in the interval of $[0, 1]$, while comparing multiple features, it is useful that each feature has its own similarity function with the same output value range. Then, each value can be weighted such that the total weighted sum is in the same value range [0,1] too.

Consider the most recent blob in a track, B_a and a detected blob B_b:

For the blobs centroid to be considered similar, they must be close in terms of their spatial position. Thus, a good centroid similarity measure is:

$$Sim_C(B_a, B_b) = 1 - \frac{EuclideanDist(C(B_a), C(B_b))}{Dist_{max}}$$

Where $Dist_{max}$ is the length of the frame diagonal (the maximum euclidean distance two objects can appear in a frame).

Two areas are similar if dividing the smaller area by the larger area results in a value closer to 1. So a good similarity function for the blob areas is:

$$Sim_A(B_a, B_b) = \begin{cases} \frac{A(B_a)}{A(B_b)}, \text{if } A(B_a) \leq A(B_b) \\ \frac{A(B_b)}{A(B_a)}, \text{otherwise} \end{cases}$$

As for the RGB histograms, a good similarity measure is to use the Bhattacharyya coefficient (in a similar fashion to [6]) which estimates the overlap between two distributions. However, since underwater images tend to have a blue or green tint, it requires a change to the approach that considers all color channels equally. Instead, three values (w_r, w_g and w_b) will control the weight each channel has in the similarity function. Considering the n binned, normalized histograms H_c for each channel c, for blobs A and B the similarity function is:

$$Sim_H(A, B) = \sum_{c \in \{r,g,b\}} w_c * \sum_{k=0}^{n} \sqrt{H_{c,k}(A) * H_{c,k}(B)}$$

Once the three feature similarity functions have been computed, similarity between blobs can be computed as:

$$Sim(A, B) = W_C * Sim_C(A, B) + W_A * Sim_A(A, B) + W_H * SimH(A, B)$$

where, once again, each similarity function is weighted by its respective weight parameter. This allows changing the importance from one metric to another as necessary. Similar blobs (such as the ones resulting from the same object in consecutive frames) will result in a higher value of their similarity function.

In order to match detected objects to existing tracks, consider the set of active tracks T, the set of detected blobs D. First, to match blobs to their respective tracks, the matrix M_C is built, where the cells in row i and column j are equal $Sim(T_i, D_j)$. Then the cells (i, j) that have the minimum value for their respective row and column, as well as having a value $\geq Sim_{min}$, are flagged as a match. This means that the i^{th} track is matched with the j^{th} detected blob.

We are now, possibly, left with a set of unmatched tracks and a set of unmatched blobs. There are four different situations that these can arise from:

1. A track whose object has left the frame, or is occluded behind the background (such as a rock).
2. A new object has appeared in the frame.
3. Two distinct objects overlapped in paths, and are detected as a single blob.
4. Previously overlapping objects have split up, and are now detected as two different blobs.

It is fair to assume that a track without a blob close to it means that situation 1. has occured. Thus, tracks whose Sim_C values with the remaining unmatched blobs are all bellow a threshold $C_{TrackReach}$ can be discarded. The analogous situation can be exploited for unmatched objects in situation 2. If they are too far from any previous tracks they are assumed to be new objects appearing in the scene and a new track is created for them.

Only the more complex overlapping situations 3. and 4. are left to deal with. When two tracked fish overlap their last centroid positions must be relatively close to each other. Similarly, when a track that refers to a group of overlapping fish is unmatched, two unmatched blobs should appear nearby. Thus, a relatively good way to check if a tracked group has split into two blobs should be to check the Sim_C for a track and each blob in unique pairings of remaining unmatched blobs.

Consider track T_a, and detected blobs B_1 and B_2. If

$$Sim_C(T_a, B_1) * Sim_C(T_a, B_2)$$

is relatively high it means that both values are also relatively high which in turn means that both blobs are close to the track. Thus, if that value is above C_{split} for any considered matching between a track and a pair of blobs then that track is split into two tracks, one for each object. This step handles situation 4. Situation 3. can be handled in a similar fashion, with pairings of tracks being matched to a single blob and resulting in a new grouped track.

After considering all possible situations mentioned above any unmatched objects are assigned a new track and any unmatched tracks are forgotten. One issue with the way splitting/merging tracks are handled as explained above is that it does not take into account how big an object appears to be in the frame. If that turns to be a problem the Sim_C should be adjusted to take that into account. Sim_A could also be taken into account if necessary, as two merging objects should appear bigger than any of the previous blobs alone, and the opposite should be true for splitting blobs.

Additionally, when a merged track is split up again the previously merged tracks can resume tracking their corresponding objects by comparing the Sim_H value between pairings.

With every case handled and no unmatched blobs or tracks remaining the tracking module outputs the updated set of tracks into the classification module.

3.5 Classification Module

As previously discussed, in the related work analysis chapter, a good approach to avoid volatile classifications, where an object might be briefly misclassified, is to use temporal consistency to our advantage. Thus, given a classification algorithm, C, for a given blob (and its track) the classification module will work as follows.

Each track will have a blob classification history H_{Class}, which contains the classes obtained from C for each blob detected and associated with that track, at each frame. Then, the track will be classified as the most frequent class in its H_{Class}, which is equivalent to the peak of its classification histogram. To avoid committing to wrong classifications early on (when a track is recent and thus has few observations in its history) the track classification should only be done when a considerable amount of observations are available.

As for C there are a few problems that deeply impact which approaches are not only feasible but also efficient. One is the variable lighting. If the lighting is similar in the video stream and the provided training data set then this problem is mostly mitigated and a simple RGB histogram clustering comparison should suffice. Another issue is the amount of diversity, or lack of it, among present fish species features (e.g. if all the species have very distinct body shapes) which would make classification more easily achievable using edge detection techniques.

Thus we would like not to commit to a specific approach for the implementation for C, but rather experiment with some of the reviewed approaches and see which performs better for our problem in particular.

4 Evaluation Method

To formulate an evaluation method for this project it is important to take into account its ultimate goals. Since the aim is to provide reliable tracking for each fish, as well as their respective classification (species, or grouped species depending on the case) it is only fair that the evaluation should be based on metrics related to these.

First, example sequences should be manually labeled with tracks for each fish, as well as their corresponding class. This is to be taken as the ground truth for the system. Then two different metrics, R_t and R_c are considered for each track. R_t is the relative number of frames where the system successfully identified the bounding box for a given fish. A bounding box is said to be successfully identified if the area overlap ratio between the ground truth bounding box and the output of the system is above a certain threshold. R_c is the ratio of frames of a sequence where a track was correctly classified. These two metrics should give a good idea of well the system is performing and can be used for further analysis.

5 Schedule

In figure 5 we outline the schedule of work for this thesis. During the development stage the proposal, in section 3, should be implemented. Then, during the case study stage, the implementation is tested. Any necessary modifications to our proposal should be made during the previous two stages. Next stage is the result evaluation where an analysis of the results from the previous stage is formulated. The next stage should be writing the article regarding our work. The thesis document is fleshed out during every previous stage and the last few weeks should be spent writing it.

6 Conclusions

In this project-thesis research was done towards the goal of providing real-time detection, tracking and classification of fish in video streams. Previous research work with similar goals or that share parts of the problem was reviewed and analyzed. Appropriate techniques were chosen and adjusted towards an approach for this problem, which is suggested in our proposal.

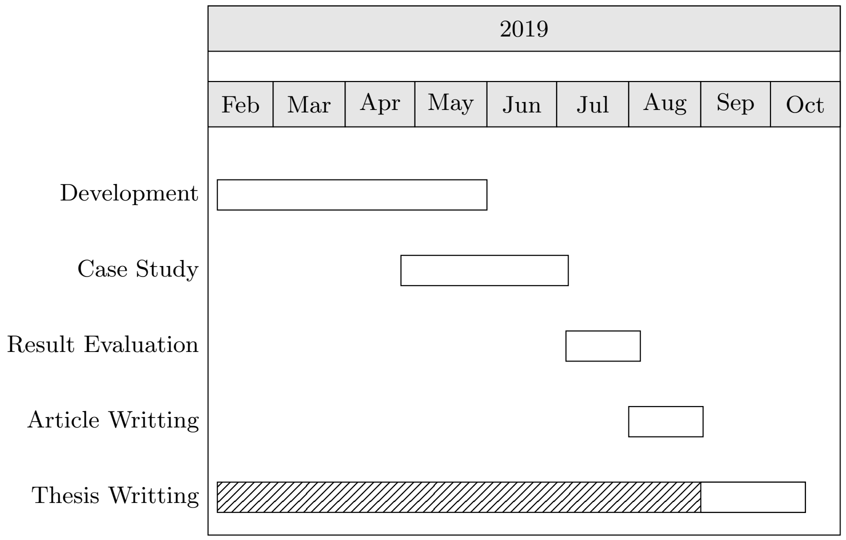

Fig. 5: Thesis work schedule.

The biggest hurdle faced was taking the real-time constraint into consideration while attempting to solve all the problems associated with our project. When keeping a balance between performance and computational weight there are issues that are hard address. The range in domains involved also made it harder to go in-depth on every topic. Thus, there are still improvements that can be done to our proposal.

References

1. Aherne, F.J., Thacker, N.A., Rockett, P.I.: The bhattacharyya metric as an absolute similarity measure for frequency coded data. Kybernetika 34(4), 363–368 (1998)
2. Bobick, A., Davis, J.: Real-time recognition of activity using temporal templates. In: Applications of Computer Vision, 1996. WACV'96. pp. 39–42. IEEE (1996)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. IEEE Computer Society Conference. vol. 1, pp. 886–893. IEEE (2005)
4. Friedman, N., Russell, S.: Image segmentation in video sequences: A probabilistic approach. In: Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. pp. 175–181. Morgan Kaufmann Publishers (1997)
5. Harvey, A.C.: Forecasting, structural time series models and the Kalman filter. Cambridge university press (1990)
6. Johnsen, S., Tews, A.: Real-time object tracking and classification using a static camera. In: IEEE International Conference on Robotics and Automation, workshop on People Detection and Tracking (2009)

7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
8. Lipton, A.J., Fujiyoshi, H., Patil, R.S.: Moving target classification and tracking from real-time video. In: Proceedings Fourth IEEE Workshop on Applications of Computer Vision. pp. 8–14. IEEE (1998)
9. McFarlane, N.J., Schofield, C.P.: Segmentation and tracking of piglets in images. Machine vision and applications 8(3), 187–193 (1995)
10. Spampinato, C., Chen-Burger, Y., Nadarajan, G., Fisher, R.B.: Detecting, tracking and counting fish in low quality unconstrained underwater videos. In: Third International Conference on Computer Vision Theory and Applications. vol. 2, pp. 514–519. INSTICC - Institute for Systems and Technologies of Information, Control and Communication (2008)
11. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Computer Vision and Pattern Recognition 1999. IEEE (1999)
12. Suykens, J.A., Vandewalle, J.: Least squares support vector machine classifiers. Neural processing letters 9(3), 293–300 (1999)
13. Villon, S., Chaumont, M., Subsol, G., Villéger, S., Claverie, T., Mouillot, D.: Coral reef fish detection and recognition in underwater videos by supervised machine learning: Comparison between deep learning and hog+ svm methods. In: International Conference on Advanced Concepts for Intelligent Vision Systems. pp. 160–171. Springer (2016)