# Comparison of Vision-based and CNN-based Classifers for Fish Monitoring in Complex Environment

**Ling Yi Jun**[1] and **Lau Phooi Yee**[2]

**ABSTRACT:** Aquaculture farming can help soften the environmental impact of overfishing by fulfilling seafood demands with farmed fish. However, maintaining large scale farms can be challenging, even with the help of underwater cameras affixed in farm cages, because there are hours' worth of footage to sift through, which can be a laborious task if performed manually. A vision-based system therefore could be deployed to automatically filter useful information from video footage. This work proposes to solve the above-mentioned problem using 2 methods for fish detection: 1) Extended UTAR Aquaculture Farm Fish Monitoring System Framework (UFFMS), a handcrafted method, and 2) Faster Region Convolutional Neural Network (Faster R-CNN), a CNN-based method. These two methods extract information about fish from video footage. Experimental results show that for well-lit footage, Faster R-CNN performs better than the extended-UFFMS. However, the accuracy of Faster R-CNN drops drastically for poorly lit footage, at an average of 28.57%, despite still having perfect precision scores. The average accuracy of the extended-UFFMS and Faster R-CNN methods are 57.89% and 71.77% respectively.

**Keywords:** Underwater Video Analysis, Machine Learning, Computer Vision, Object Detection, Fish Monitoring

**DOI:** 10.37936/ecti-cit.2021152.240265

**Article history:** received March 28, 2020; revised May 20, 2020; accepted June 28, 2020; available online April 30, 2021

## 1. INTRODUCTION

The global demands for seafood have led to the fishing-down phenomena. This has impacted the base of the food chain, causing a decline in the entire marine food web [1]. Industrial-scale fishing farms or aquaculture farms can stem overfishing in the long run. Due to the various regulatory frameworks for aquaculture, where monitoring is obligatory, various monitoring systems are deployed to measure the quality of fish and farming conditions [2]. However, maintaining 24/7 monitoring systems for large scale farms requires a lot of manpower to collect information and analyse it. In response to this, vision-based systems are used to assist the fish farm industry in reducing their operational workload in monitoring fish by automating the monitoring process. However, the data collected from these vision-based systems requires manpower to sift through hours' worth of underwater footage in order to extract meaningful information such as the quantity and the quality of their production. Therefore, most aquaculture farms require some form of automatic system to process acquired images. Finding a better way to do this is becoming a pressing issue. An automatic vision-based monitoring system can reduce production cost, which could lower the retail prices, thus making farmed fish an affordable choice for consumers.

In 2018, Lukežič et al [3] introduced a tracking algorithm which utilizes channel and spatial reliability concepts for discriminative correlation filters (DCF), resulting in enlarged search regions, which are useful for the tracking of odd-shaped objects. In 2015, Tan et al [4] proposed tracking both lobsters, using a particle filter-based method, and burrows, using Lucas-Kanade's optical flow method, because the lobsters move freely under the moving camera. Later in 2018, Tan et al. [5], proposed an automatic approach to estimate the abundance of Norway lobsters. In 2014, Fier et al [6] proposed using a handcrafted algorithm to detect fish in highly turbid environments which

[1,2]The authors are with the Faculty of Information and Technology, Tunku Abdul Rahman University, Malaysia., E-mail: : lingyj80@1utar.my and laupy@utar.edu.my

***Table 1:*** *Description of video footage used in experiments.*

| Video | Video Information | Description of the Video | Complexity | Fish farms that relate to conditions depicted in the footages |
|---|---|---|---|---|
| **(a)** | 596 × 336 (50fps) (15s) | • **Grey reef sharks in backlight [12]**<br>• Camera looking upwards<br>• Salient object in backlight condition<br>• Dimly illuminated | • Foreground objects have similar color to the background<br>• Stark illumination gradient present | • Mediterranean (island of Ibeza, Spain) Bluefin Tuna Farm<br>• Open offshore cage, supported with heavy mesh net, 49 meter diameter farm cage [15] |
| **(b)** | 596 × 336 (24fps) (29s) | • **Large school of fish swimming in the Red Sea in Egypt [13]**<br>• Camera looking sideways<br>• Salient object in flat light condition<br>• Brightly illuminated | • Background noise (coral, reflection of water) present<br>• Illumination gradient present | • Hong Kong (Lau Fau Shan) Giant Grouper Farm<br>• Indoor recirculating aquaculture system, with artificial lighting [16] |
| **(c)** | 596 × 336 (30fps) (11s) | • **School of fish blue background [14]**<br>• Camera looking sideways<br>• Salient object in butterfly light condition<br>• Brightly illuminated | • Illumination gradient present | |

involves using 2 background subtraction techniques to utilize the strengths of each technique. Both of the background subtraction algorithms are computed independently, and then later combined with logical operations

Due to the success of CNN-based methods for object detection, some researchers have started to utilize deep learning methods in fish monitoring systems. In 2016, Villon et al [7] detected fish in an unconstrained environment using the GoogleNet [8] architecture with 27 layers, 9 inception layers and a soft-max classifier. The inception modules allow the dimension of the picture to be reduced to one pixel, solving the over-fitting issue and reducing computational expense. Jäger et al [9] proposed a method where the novelty lies in the generation of initial bounding box recommendations, instead of the usual sliding window or selective search for object localization in deep learning networks. The original footage goes through a background subtraction algorithm, resulting in a binary mask. Erosion is applied to the binary mask to separate fish that are close together, resulting in a secondary mask. Then, a blob detection algorithm is used to detect blobs from both binary and secondary masks to generate initial bounding boxes, with boxes less than 100 pixels removed due to them being too small for classification tasks. In 2018, Mandal et al [10] experimented on the region proposal network (RPN) found in Faster R-CNN, and combined it with three different CNN-based classification models with different sizes (small, medium and large) to obtain Faster R-CNN models of three different sizes (i.e. the number of layers), and compared each network's performance. In 2019, Ling et al. [11] proposed a hand-crafted method to automatically count the number of fish in underwater video footage. Each frame from the video footage will go

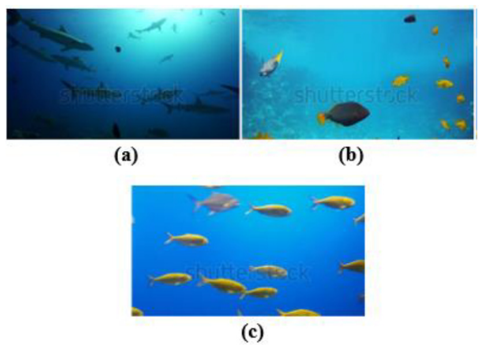


***Fig.1:*** *(a) Grey reef sharks in backlight; (b) A large school of fish swimming in the Red Sea in Egypt;(c) School fish blue background.*

## 2. EXTENDED UFFMS FRAMEWORK

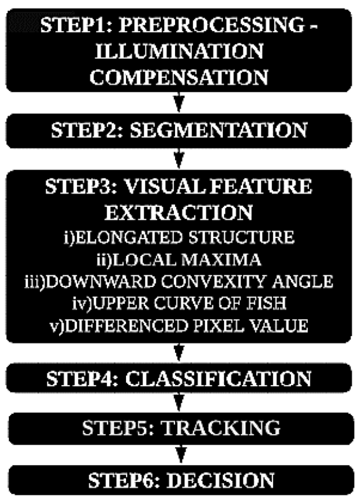

STEP1: PREPROCESSING - ILLUMINATION COMPENSATION
STEP2: SEGMENTATION
STEP3: VISUAL FEATURE EXTRACTION
i)ELONGATED STRUCTURE
ii)LOCAL MAXIMA
iii)DOWNWARD CONVEXITY ANGLE
iv)UPPER CURVE OF FISH
v)DIFFERENCED PIXEL VALUE
STEP4: CLASSIFICATION
STEP5: TRACKING
STEP6: DECISION

***Fig.2:*** *Extended-UFFMS Framework.*

through four modules: 1) pre-processing, 2) segmentation, 3) classification, and 4) decision. Therefore, in this paper, the goals of Ling are revisited with additional features considered, and this extended framework is further compared with CNN-based methods.

The remainder of this paper is organized as follows. In Section 2 we describe the characteristic of the video sequences. Subsequently, section 3 describes the extended-UFFMS framework which extends the previous research's (UFFMS) framework with: 1) an addition of a tracking module, and 2) an improvement of a visual feature in the classification module. Section 4 discusses the CNN-based method used to detect fish in an underwater environment with an overview of different CNN object detection architectures. Section 5 shows the experimental results with discussions and comparisons. Lastly, section 6 concludes the paper and suggests possible future work.

## 3. VIDEO SEQUENCE CHARACTERISTICS

The video sequences were obtained online from varied environments. These videos are related to the following conditions: 1) indoor recirculating aquaculture systems, 2) open offshore cages, or 3) large schools of fish swimming in the ocean – see Table 1. The sample frames can be seen in Fig 1, while the details and descriptions of each video can be found in Table 1. Due to having light sources coming from only one direction, which was from above water, a consistent issue found in the sample frames is uneven illumination.

The UFFMS framework is a non-intrusive method to monitor fish. Specifically, it is used to count and calculate the sizes of underwater fish in varying environments, especially where illumination is non-even across the frame [11]. However, the method proposed in the UFFMS framework could not (1) count occluded fish, or (2) differentiate certain non-fish foreground objects due to ineffective visual features (eg. coral is misidentified as fish). In order to decrease the number of false negatives in detection, which is mainly due to the failure to segment fish, the extended UTAR Aquaculture Farm Fish Monitoring System (UFFMS) framework, has a 1) modification for Step 3: $(ii)$ Local Maxima, and 2) adds a tracking module, Step 5: Tracking – see Fig. 2. Individual frames from video footage are extracted and processed individually. Each selected frame will go through the five main processing modules, namely: preprocessing, segmentation, classification, tracking, and decision.

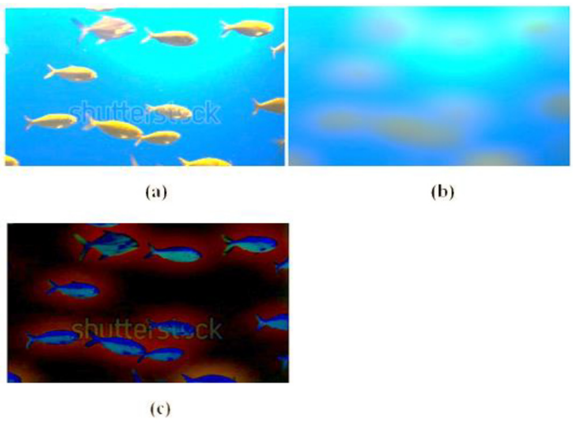


***Fig.3:*** *(a)Acquired Frame; (b)Gaussian Mask; (c)Subtracted Image.*

A pre-processing method is employed to correct the uneven lighting issue. First, Gaussian blur [17] is used to create a Gaussian Mask, which defines the gradient of illumination, also known as the illumination descriptor, see - Fig 3(b). The Gaussian Mask is generated by blurring out the foreground objects (the fish) of the acquired frame and retaining the uneven background illumination. The equation of Gaussian Blur is given in equation 1:

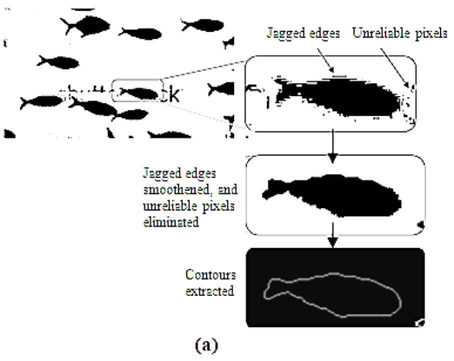
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\dfrac{x^2+y^2}{2\sigma^2}} \qquad (1)$$

$x$ represents the distance from the origin in the x-axis, while $y$ represents the distance from the origin in the y-axis. $\sigma$ stands for the standard deviation of the Gaussian distribution.

Then, the Gaussian Mask, which represents the background, is subtracted from the Acquired Frame. This process is also known as background subtraction. The result of the background subtraction is an evenly illuminated frame with the foreground object clear to observe, as can be seen in Fig 3(c), Subtracted Image.

### 3.1 Step 2: Segmentation

The purpose of this step is to separate foreground objects from the background. In Fig 3(c), along the edges of the foreground objects there are reddish black "smears" due to the blurring of Figure 3(a). This is because, during the blurring a process, any rapid change in pixel intensity that occurs from the edge of the fish to the background is averaged out, causing the pixels outside of the contour to have a pixel intensity that represents the foreground object. Therefore, thresholding is applied on Fig 3(c) to remove the "smears", and to convert the frame into a binary image, with white pixels representing the background and black pixels representing the foreground objects.



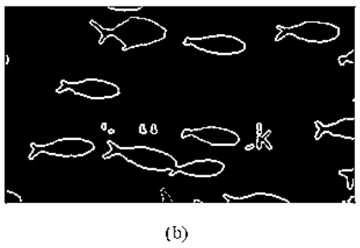


*Fig.4:* *(a)top- Result of thresholding with jagged edges and unreliable pixel present;(a)middle-morphological closing result; (a)bottom-contour extraction result; (b) resulting frame after contour extraction.*

The result from the thresholding process are "blobs" that have jagged edges. Unreliable and noisy pixels are also present throughout the frame – see Fig4 (a) top. Therefore, to remove jagged edges of the "blobs" and to eliminate unreliable and noisy pixels, morphological closing is applied. The result after morphological closing can be seen in Fig 4(a) middle, where the edges of the blobs are smooth and the noisy pixels are eliminated.

Then, contours are extracted from the binary frame. Contours define edge information for the foreground objects, which is needed for the next module- Feature Extraction. An edge-based contour extraction algorithm [18] is applied to define each object's contours. The result can be seen in Fig 4(b), where the 2D shape of the fish is clearly defined in the form of a closed contour.

## 3.2 Step 3: Feature Extraction

In Step 3, five visual features are extracted to determine whether a contour represents a fish object. These five visual features are: **F1)** Elongated Structure, **F2)** Local Maxima, **F3)** Downward Convexity Angle, **F4)** Upper Curve of Fish, and **F5)** Differenced Pixel Value. Note that in the previous UFFMS framework, feature 2 (**F2**) is Downward Curve of Fish, while in this extended-UFFMS framework, this feature has been improved into Local Maxima. The following paragraphs explain each feature.

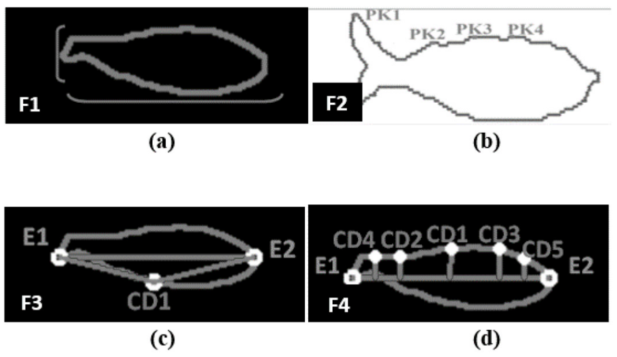


***Fig.5:*** *(a)* ***F1*** *– Elongated Structure; (b)* ***F2*** *– Local Maxima output; (c)* ***F3*** *– Downward Convexity Angle output; (d)* ***F4*** *– Upward Curve of Fish output.*

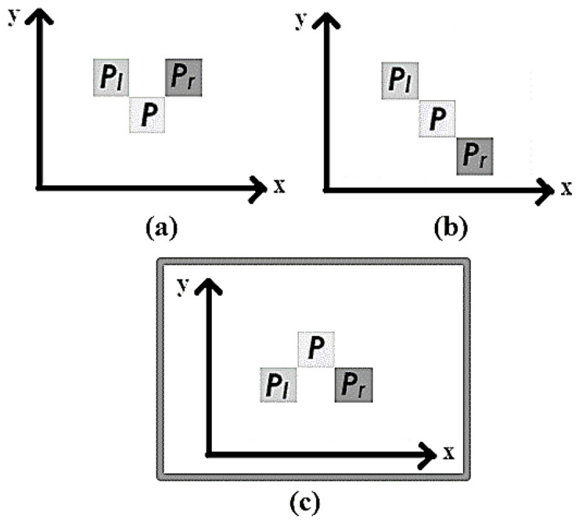


***Fig.6:*** ***F2*** *– Local Maxima explanation,* ***P****= y-coordinate of current pixel,* ***Pr*** *= y coordinate of right connected pixel,* ***Pl*** *= y-coordinate of left connected pixel; (a) Example of non-peak; (b) Example of non-peak; (c) Example of a peak.*

**F1** – Elongated Structure: Usually, the structure of a fish has an elongated shape. In order to decide if a contour has an elongated structure, the system first creates a bounding box around the contour. Then, the height and width of the bounding box are compared. If one side of the bounding box is at least twice the length of the other, regardless of which side, the contour is said to have matched the Elongated Structure visual feature - see Fig 5(a).

**F2** – Local Maxima: This feature in the previous framework was called Downward Curve of Fish, which only determined if part of the contour is curved. In this extended framework, it is improved to find the Local Maxima, which detects peaks in a curvature. Local Maxima detects peaks by comparing the y-coordinate of each pixel in a curve of a contour. If the y-coordinate of a particular pixel is higher than the y-coordinate of both its adjacent left and right pixels, that particular pixel can be considered a peak. An example of a peak can be seen in Fig 6(c). This process is repeated until every pixel of the curve on the contour has been iterated. If a detected curve has less than 4 peaks, the contour is considered to have matched with this visual feature -see Fig 5(b).

This Local Maxima visual feature enables the system to eliminate noise caused by imperfect segmentation (eg. coral reefs, water reflection). Those erroneous contours have more than 4 protrusions (peaks), as can be seen in Fig 7.

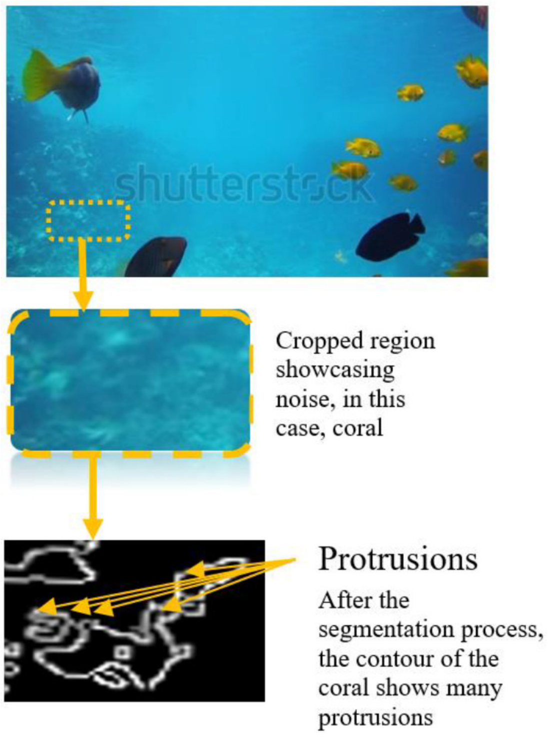

shutterstock
Cropped region showcasing noise, in this case, coral
Protrusions
After the segmentation process, the contour of the coral shows many protrusions

***Fig.7:*** *top - Sample original frame; middle-ROI with sample noise, in this case, coral; bottom-Contour of the coral.*

**F3** – Downward Convexity Angle: Due to the position of a fish's belly, the shape of a fish consists of a downward convex from the fish's contour. It is noted that this visual feature is not useful for detecting fish with thin proportions, such as Barracuda. This visual feature extracts 3 points from the contour in the beginning, E1, E2, and CD1, - see Fig 5(c). A triangle is drawn by connecting those 3 points with 3 separate lines. The resulting angle formed at CD1 is then calculated with equations 2 and 3

$$\cos\theta = \frac{\bar{a}\cdot\bar{b}}{\|\bar{a}\|\cdot\|\bar{b}\|} \tag{2}$$

$$\begin{aligned}\bar{a} &= \overline{E1} - \overline{CD1}\\ \bar{b} &= \overline{E2} - \overline{CD1}\end{aligned} \tag{3}$$

$\Theta$ represents angle of CD1. The contour is said to have matched the visual feature when equation 4 is satisfied.

$$f3 = \theta_L < \theta < \theta_H \tag{4}$$

**F4** – Upper Curve of Fish: The purpose of this feature is to detect whether or not the shape of the upper part of the fish is curved. First, points E1 and E2 are plotted in the middle of the shorter edge of the bounding box. Then a line is drawn to connect the two points. This line, also known as the equator line, cuts the fish horizontally in half. Five points, arranged from left to right, P4,2,1,3,5, are then assigned on the Equator Line. After that, for each plotted point on the Equator line, a direct perpendicular line is drawn in an upwards direction until the contour is reached. The points where each line intersects with the contours are assigned as points, from left to right, CD4,2,1,3,5 respectively. This can be seen in Fig 5(d). The lengths of each perpendicular line are measured to compute the standard deviation. If the standard deviation is between 2 and 50, the contour matches with the visual feature, as it shows that the upper curve of a contour is indeed a curve.

**F5** – Differenced Pixel Value. The average colour value for the region of interest (ROI) which contains a fish is different from the average colour value for the whole input image. This can be seen in Fig 8 and 9.

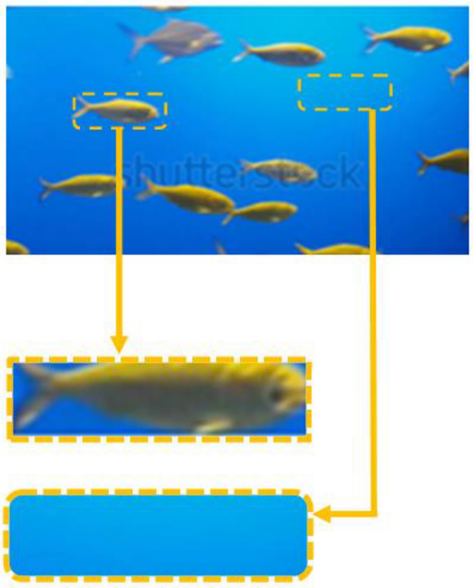


***Fig.8:*** *top-Acquired Frame; middle- ROI with fish; bottom- ROI without fish.*

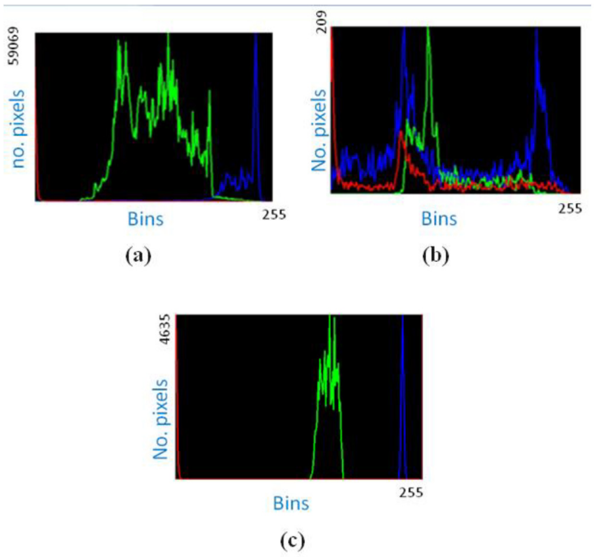


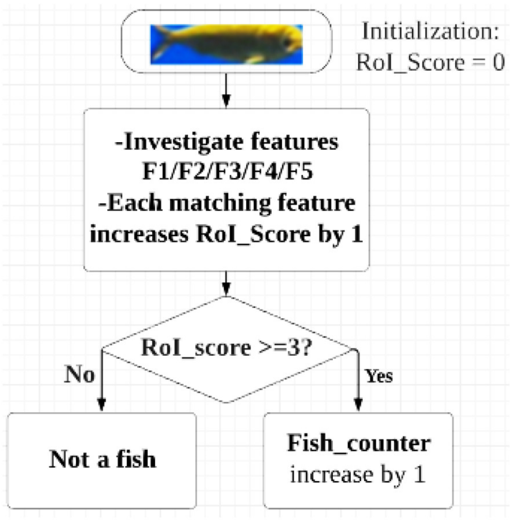
***Fig.9:*** *(a)Acquired Frame colour histogram; (b) ROI with fish colour histogram; (c)ROI without fish colour histogram.*

For the green colour value, the acquired frame has a wider range of pixel intensity values compared to the cropped ROI. The colour values of the ROI containing a single fish and the acquired frame are averaged and compared. If the averaged values exceed a certain threshold value, in this case 20 which was obtained after vigorous testing, the contour is said to match this visual feature. The calculation of averaged values can be summarized in equation 5, with R, G, B representing the intensity values of the red, green, and blue pixels respectively, and the 'ori' subscript representing acquired frame, while the 'roi' subscript represents region of interest, which is the region encapsulated by each contour's bounding box.

$$Ave_{CV} = \frac{\begin{array}{l} Diff(R_{ori}, R_{roi}) + Diff(G_{ori}, G_{roi}) \\ +Diff(B_{ori}, B_{roi}) \end{array}}{3} \tag{5}$$

## 3.3 Step 4: Classification

In the classification step, for each contour, a counter named *RoI_Score* is initialized with value 0. The purpose of the counter is to keep track of how many visual features the particular contour matches. Each contour will go through five visual features. For each feature that the contour matches, the *RoI_Score* will be incremented by 1. After all of the 5 visual features have been tested, if *RoI_Score* is more than 3, the contour is considered a fish and the results are recorded. Through rigorous testing, the *RoI_Score* was set to 3, as any lower value caused more false positives, and higher values caused more false negatives. A flowchart representing the classification process is shown in Fig 10.



Initialization:
RoI_Score = 0
-Investigate features F1/F2/F3/F4/F5
-Each matching feature increases RoI_Score by 1
RoI_score >=3?
No
Yes
Not a fish
Fish_counter increase by 1

***Fig.10:*** *Classification flowchart.*

## 3.4 Step 5: Tracking

The previous UFFMS framework has an issue with occluding fish. When fish occlude one another, the system will detect multiple fish as one big fish. This is due to the edge-based contour detector which only outlines the outer most edge of the contour, as can be seen in Fig 11(b).

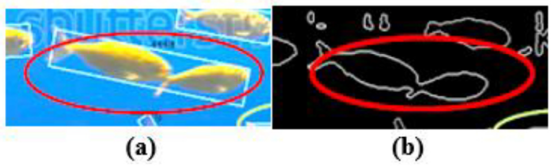


***Fig.11:*** *(a)Sample output; (b) Debug frame.*

To solve this issue, a tracking algorithm called Discriminative Correlation Filter with Channel and Spatial Reliability (DCF-CSR), from Lukežič et al [3], is deployed. A brief explanation of DCF-CSR follows. Channel Reliability weights and a Spatial Reliability map are used to seamlessly integrate filter updates with the tracking process. The Spatial Reliability map is used to adjust filter support to the part of the selected region from the frame for tracking, while the Channel Reliability weights reduce noise of weight averaged filter responses. A more detailed explanation can be found in Lukežič et al's [3] paper. The tracker initialization is shown in Fig. 12. The pseudo code for tracker initialization is shown in Fig.13. The coordinates from the minimum bounding box of the contour are used to determine if a tracker already exists for a particular contour. If no tracker exists, a new tracker is initialized and assigned to that particular contour. If a tracker already exists, no action is performed.

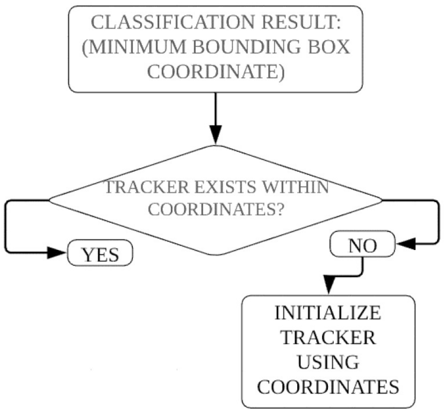

CLASSIFICATION RESULT: (MINIMUM BOUNDING BOX COORDINATE)
TRACKER EXISTS WITHIN COORDINATES?
YES
NO
INITIALIZE TRACKER USING COORDINATES

***Fig.12:*** *Tracker initialization flowchart.*

```
If ((frame%5) = = 0) {
    Detection module classifies existing contours
    For (all detected contours) {
        If (tracks array! =0) {
            For (all tracks) {
                If (Detected contour coordinates! = existing track coordinates) {
                    New track is initialized at detected contour coordinates}
                }
            }
        }
}
Else {
    For (all existing tracks) {
        Predict new location and size of track using DCF-CSR}
}
```

***Fig.13:*** *Tracker initialization pseudo code.*

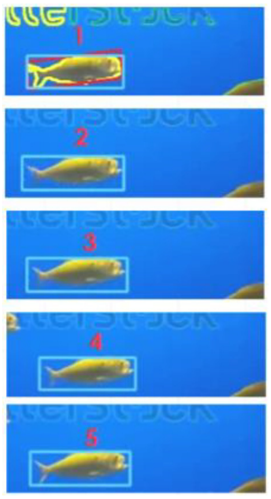


***Fig.14:*** *A fish successfully tracked for five consecutive frames.*

## 3.5 Step 6: Decision

A decision that a particular contour is a fish is made when the tracker successfully tracks the fish for five frames consecutively. Fig.14 shows a series of five frames depicting a successfully consecutively tracked fish. In contrast, Fig. 15 depicts a series of frames of a misclassified contour, which have not been successfully tracked consecutively for five frames.

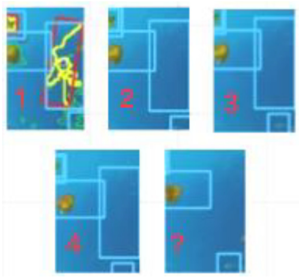


***Fig.15:*** *Unsuccessful consecutive tracking of 5 frames.*

## 4. FASTER R-CNN METHOD

### 4.1 Review of different CNN-based object detection architectures and their localization methods

Underwater fish have different aspect ratios and sizes when viewed in video footage. This makes the position of the fish for computer vision algorithms unpredictable. Also, the number of fish that appear in a frame is also inconsistent. Therefore, a localization method is needed.

**CNN:** Localization for a CNN involves "sliding windows". A window frame which has an arbitrary size smaller than of the image is created and placed at the corner of the image. Then, this window frame is fed into a CNN to determine if the objects exist in that area of the image. The window frame is then moved to the next part to detect if an object exists or not. This process is repeated for the entire image. The "sliding window" method is computationally intensive.

**R-CNN:** R-CNN [19] is an improvement from the sliding window method, with a selective search algorithm [20] responsible for selecting 2000 regions from the image, also known as region proposals. Although this method is an improvement compared to the sliding window method, it is still computationally expensive to classify 2000 regions per frame.

**Fast R-CNN:** Fast R-CNN [21] is an improvement of R-CNN. The method is similar to the R-CNN algorithm, but the region proposals are not fed into the CNN. Instead, the original input image is fed into the CNN to get the convolutional feature map. From the convolutional feature map, only the region proposals identified using selective search are selected and warped into squares before being fed into a fully connected layer.

**Faster R-CNN:** A neural network proposed by Ren et al [22], named Faster R-CNN, is used to detect fish in the footage. As the name implies, it is faster than Fast R-CNN. Similar to Fast R-CNN, the input image is first fed into the convolutional network which outputs a convolutional feature map. However, instead of deploying a selective search algorithm on the feature map to extract region proposals, a separate network is used

**YOLO:**YOLO (You Only Look Once) [23] is fundamentally different from other algorithms in terms of its region proposals and it is much faster. However, the downside is that it is not suitable for detecting relatively small objects, due to the spatial constraints of the algorithm when determining the size of the grid on input. YOLO is not suitable because underwater fish can sometimes be relatively small.

**SSD** (Single Shot Detectors) [24] are faster than Faster R-CNN and YOLO. (Inference speed of Faster R-CNN = 425ms, SSD = 89 ms). This is achieved by predicting bounding boxes after multiple convolutional layers. However, the accuracy of SSD was not acceptable when tested on our dataset, even when the minimum score threshold had been reduced to as low as 0.05. Examples can be seen in Fig 16.

Faster R-CNN's RPN network enabled it to be faster than its predecessors, R-CNN and Fast R-CNN, which use selective search as the region proposal algorithm, without compromising accuracy. SSD and YOLO, on the other hand, are faster than Faster R-CNN, but their accuracy could be compromised when object sizes are small. This is because SSD predicts box offsets from default bounding boxes by applying small convolutional filters to feature maps. This process reduces the spatial dimension and resolution, resulting in the high error rate when SSD is used for detecting smaller objects. As for YOLO, due to the spatial constraints of the algorithm when determining the size of the grid on input, this method is also not suitable to detect smaller objects. Therefore, due to the fish's unpredictable aspect ratios and sizes, which can sometimes appear small, the Faster R-CNN architecture was chosen for its better accuracy, despite longer inference time when compared to YOLO and SSD.

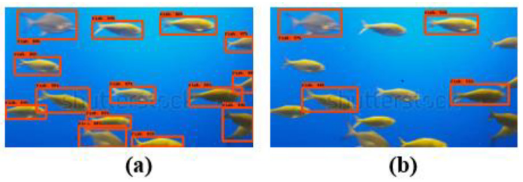


***Fig.16:*** *(a) Sample detection result of Faster R-CNN; (b)Sample detection result of SSD.*

## 4.2 Faster R-CNN

Anchor boxes are important for the Faster R-CNN architecture. The default setting for Faster R-CNN has 9 anchors for a position in an image. The number of positions and the stride can be arbitrarily set. Fig. 17 [25] shows the 9 anchors at position (320,320) in an image.

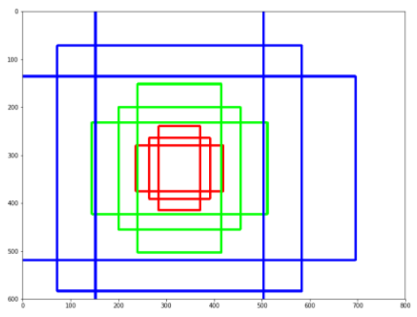


***Fig.17:*** *Anchor boxes visualization at a particular position (320,320), with height and width ratio of 1:1, 1:2, and 2:1, in an image of size (600, 800).*

Three colors represents three different scales: 128 × 128, 256 × 256, and 512 × 512. The anchors have height to width ratios of 1:1, 1:2, and 2:1. The computational resources needed to classify all the spawned anchor boxes are similar to that of sliding windows. However, the region proposal network (RPN) shares computational resources with the object detection network. RPN predicts the probability of an anchor box being a background or a foreground, and refines the anchor. Therefore, RPN greatly reduces the number of regions that need classifying in an image. The output of RPN is a set of proposed regions with inconsistent sizes. This means that CNN feature maps will be inconsistent as well. ROI Pooling is implemented to reduce inconsistent feature maps of the same size. This is done by splitting input feature maps into an arbitrary number of equal regions, and then Max-Pooling is applied on every region.

## 4.3 Model details

This section describes the details and hyper parameter set used when training and setting up the model. The Open Images Dataset v4 (OID v4) [26] was used for ResNet [27] model training. The dataset includes 600 classes, with 1743042 training images, of which 23195 are training images of various types of fish in different environments. A csv files containing annotations of bounding boxes and their labels of training (1743042 images), validation (41620 images), and testing (125436 images) was used. It can be downloaded from the official OID v4 website directly. ResNet does not perform well when the network depth increases, due to the vanishing gradient problem. As the gradient is back-propagated to previous layers, repeated multiplication with local gradients will make the gradient infinitely small. This causes the network performance to saturate and degrade. ResNet solves the vanishing gradient problem by performing skip connections by stacking identity mappings, which does not degrade the network performance. As a result, ResNet was able to train deep networks without the vanishing gradient problem. First of all, data preparation is needed. Random horizontal flips are applied as a data augmentation step to increase system robustness. Then the csv file, which contains annotation information, and the corresponding image files are converted into TFRecord, a binary file format that contains both annotation and image information.

Before training begins, weight initialization is needed. This is to prevent loss gradients from being too large or too small, where both cases will cause the network to take more time to converge, or in the worst case it may not converge at all. Variance scaling is used for the initializer together with the average numbers of input and output units. Thus it is capable of scaling adaptation to the shape of weights. The number of training steps was arbitrarily set at 10000000 before the model stops learning. The initial learning rate was set at 0.00006. After the 6000000th step, the learning rate was decreased to 0.000006. At the 7000000th step, the learning rate was further decreased to 0.0000006. This decrease in learning rate is made so that the network will converge faster.

In order to avoid overfitting the data during training (weight updates), an L2 regularizer, also known as Ridge Regression, is applied at the convolutional layers without weight decay. Regularizers are used to avoid the overfitting problem by adding biases through penalty. The L2 regularizer is used instead of L1 because it does not reduce features quickly. In order for the network to converge more efficiently, an optimizer named Momentum [28], is used. This is because high variance oscillations in Stochastic Gradient Descent (SGD) make convergence hard to reach. Therefore, Momentum accelerates SGD convergence by softening oscillations when it navigates in irrelevant directions.

During localization for the generation of the initial bounding boxes, aspect ratios are kept at a minimum dimension of 600 and maximum of 1024. Anchor centres have a stride of 6 for both height and width. A total of 125,436 images provided by OID v4 were used for testing. The mean average precision achieved was 54After training, the classifier can be used in the object detection architecture. A flowchart of how Faster R-CNN makes its inferences can be seen in Fig.18. The detection results from the deep learning method will later be compared with the results from the handcrafted method.

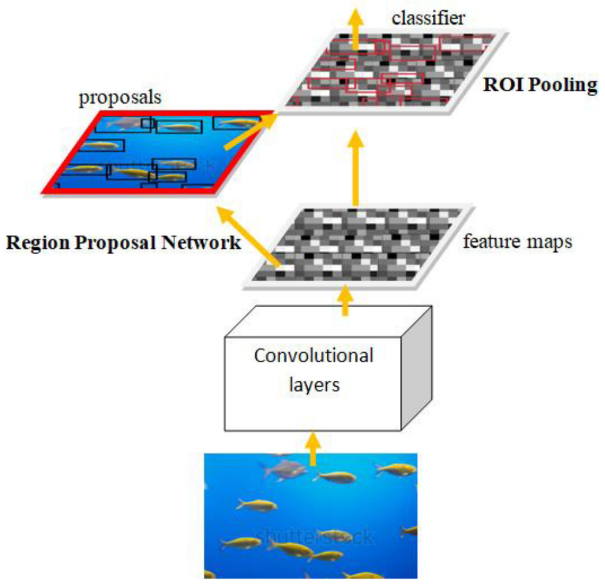


***Fig.18:*** *Steps for Faster R-CNN processing.*

## 5. EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the performance of the extended-UFFMS system, the three different sets of footage described in section 2 were used in the experiments. The implementation for both methods is similar. In both, the video footage is fed into the algorithm and bounding boxes representing a fish are drawn. For the Faster R-CNN method, it can be said that each frame is processed separately without any information from the previous frame. For the extended-UFFMS method, pixel information enclosing the bounding box is brought forward when processing the next frame to track the fish. For each of the footages' output, five frames are selected randomly to be analyzed. For each selected frame, the ground truth of the number of fish was manually counted and recorded. Note that fish in the background that are too small are not counted as ground truth. The system's output for the selected frames was recorded as well. Table 4 shows the experimental results for the extended-UFFMS framework.

***Table 2:*** *UFFMS [11] result.*

| Video | Avg (accuracy of 5 frames) (%) | Recall **(R)** (TP/TP+FN) | Precision **(P)** (TP/TP+FP) | F-score ((2×P×R)/P+R) |
|---|---|---|---|---|
| **(a)** | 63.86 | 0.86 | 0.90 | 0.88 |
| **(b)** | 76.27 | 0.90 | 0.86 | 0.88 |
| **(c)** | 78.42% | 0.90 | 0.91 | 0.91 |
| **Total** | 74.59% | 0.89 | 0.89 | 0.89 |

***Table 3:*** *Extended-UFFMS result.*

| Video | Avg (accuracy of 5 frames) (%) | Recall **(R)** (TP/TP+FN) | Precision **(P)** (TP/TP+FP) | F-score ((2×P×R)/P+R) |
|---|---|---|---|---|
| **(a)** | 53.64 | 0.64 | 0.91 | 0.73 |
| **(b)** | 53.53 | 0.70 | 0.84 | 0.75 |
| **(c)** | 66.53 | 0.69 | 0.99 | 0.79 |
| **Total** | 57.89% | 0.68 | 0.91 | 0.76 |

***Table 4:*** *Faster R-CNN result.*

| Video | Avg (accuracy of 5 frames) (%) | Recall **(R)** (TP/TP+FN) | Precision **(P)** (TP/TP+FP) | F-score ((2×P×R)/P+R) |
|---|---|---|---|---|
| **(a)** | 28.53 | 0.29 | 1.00 | 0.44 |
| **(b)** | 92.00 | 0.92 | 1.00 | 0.95 |
| **(c)** | 94.80 | 0.95 | 1.00 | 0.97 |
| **Total** | 71.77% | 0.72 | 1.00 | 0.79 |

A description of the dataset used in the experiments can be found in Section 2. According to Tables 4 and 5, deep learning method's average accuracy rate is 13.88% higher than the extended-UFFMS framework. However, the grey highlighted column shows that even though deep learning methods have a higher accuracy rate, and are able to detect occluded fish, under low light conditions as can be seen in the Grey reef sharks with backlight footage, the deep learning method's accuracy dropped by 28.57% compared to the extended-UFFMS method.

Note that for the extended-UFFMS framework, there exists an average of 10.53% error rate in the detection phase. The result of the detection phase is passed on to the tracking module, which will continuously track the wrong region, thus reducing the accuracy of the framework compared to UFFMS. However, the precision rate for the extended-UFFMS is higher than UFFMS-see Tables 3 and 4.

The black highlighted column shows that the deep learning method, using a minimum threshold score of 0.5, did not have any false positive fish count. Note that a minimum threshold score of 0.5 was used. Whereas for the extended-UFFMS network, with a total precision score of 0.91, the system sometimes erroneously classified the background as a fish.

As can be seen in the result tables, for the footage with low lighting conditions, the deep learning method does not perform well. This is due to the lack of training data that represents underwater fish in low light conditions. Training the network with datasets representing underwater fish in low lighting conditions could help improve detection results for Faster R-CNN. However, it is not an easy task to acquire large sets of such specific data.

In particular, the extended-UFFMS methods were able to detect foreground objects in footage with poor lighting (Grey reef sharks with backlight) better than Faster R-CNN. See – Fig 19.

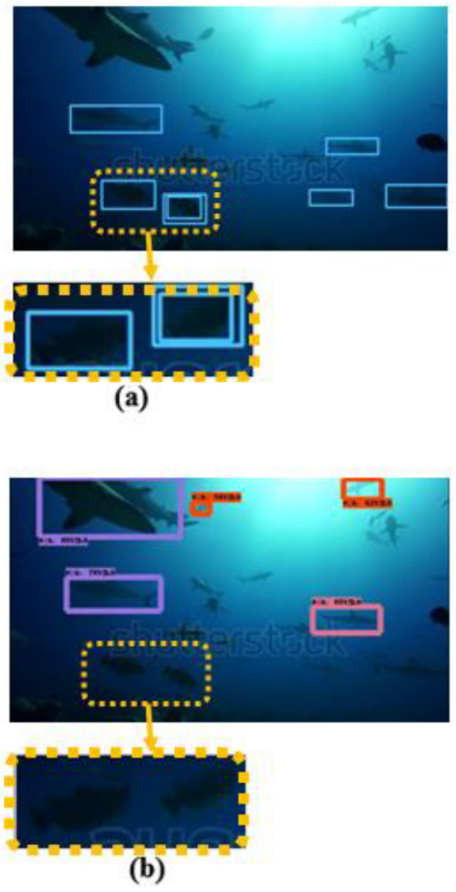


***Fig.19:*** *(a) Successful detection for extended-UFFMS framework; (b) Unsuccessful detection for Faster R-CNN method.*

The tracking module for the extended-UFFMS framework also successfully tracked and separated fish that occluded one another. This can be seen in Fig 20. Before the tracking module was implemented, the system erroneously categorized groups of occluded fish as a single big fish.

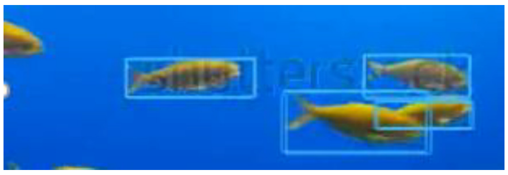


***Fig.20:*** *However, when a contour is erroneously classified as a fish, the tracker continues to track the non-fish object in the footage. An example can be seen in Fig. 21.*

However, when a contour is erroneously classified as a fish, the tracker continues to track the non-fish object in the footage. An example can be seen in Fig. 21.

***Table 5:*** *Comparison with other works.*

| Method | Scenario | Data Information | Computation (Light/Heavy) | Automatic (Yes/No) | F-Score |
|---|---|---|---|---|---|
| **Fier et al. [6]** | -Darkened appearance of fish<br>-Highly turbid environment | not known (100 1 minute video, fps not known) | Moderate (runtime NA) (Processor information NA) | Yes | 0.74 |
| **Villon et al. [7]** | -Unconstrained environment<br>-Highly textured natural background | 400 | Heavy (runtime NA) (Processor information NA) | Yes | 0.63 |
| **Faster R-CNN** | -Unconstrained environment<br>-Uneven illumination | 15 | Heavy (40.0s per frame) | Yes | 0.79 |
| **Extended-UFFMS** | -Unconstrained environment<br>-Uneven illumination | 15 | Moderate (0.8s per frame) | Yes | 0.76 |

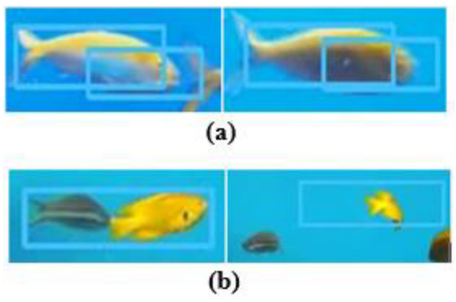


***Fig.21:*** *(a) Tracking imperfection due to imperfect segmentation of fish edges; (b) Tracking imperfection due to occluding effect of fish coinciding with the detection phase.*

Faster R-CNN had no occluding fish issue. The deep learning network was able to accurately differentiate fish instances despite fish objects being occluded. This can be seen in Fig 22.

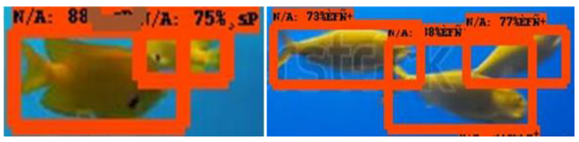


***Fig.22:*** *Successful detection of partially overlapping objects for Faster R-CNN method.*

## 6. CONCLUSIONS AND FUTURE WORK

This paper can be seen as having 2 parts: 1) Extended-UFFMS Framework, and 2) Faster R-CNN method. Both methods are non-intrusive ways of automatically monitoring underwater fish to count the number of fish in different types of environments. Such an automated system aims to benefit fish farm industries where the monitoring of the fish is frequently done manually.

In the experiments to compare the performance of the extended–UFFMS framework and the Faster R-CNN method, the Faster R-CNN method performed better for well-lit footage than the extended-UFFMS framework. The reason the accuracy rate for the extended-UFFMS framework is lower by 13.88% lies in the tracking module. Erroneously detected regions that have been passed on to the tracking module will continue to be tracked, lowering accuracy of the framework. However, accuracy of the extended-UFFMS framework is higher by 25.11% than Faster R-CNN for non-well-lit footage. Faster R-CNN was able to achieve a perfect precision score of 1, which means there were no false positives in detecting fish.

A comparison for a few fish monitoring methods can be seen in Table 6. Experiments for the Faster R-CNN method were run using Google Collab's default processor, which is the default-n1-highmen-1 instance, 2vCPU@2.2GHz, with 13GB RAM. The extended-UFFMS framework was run on Intel® Core$^{TM}$ i5-4210U 1.7GHz with Turbo Boost up to 2.7GHz, with 4GB DDR3 L Memory. The average time taken to process a frame for the extended-UFFMS framework was 0.8s, which is 50 times faster than the Faster R-CNN method, at an average of 40.0s. However, note that different processors are used. The extended-UFFMS method's main computation lies in the visual feature matching section. For Fier et al's method [6], the main computation lies in the two background subtraction techniques that are computed independently. The F-score for Faster R-CNN method is highest among all of the fish monitoring methods, at 0.79. However, note that the datasets used for the fish monitoring methods are different. The challenge faced when using Villon et al's [7] dataset lies in the highly textured background, which is not prevalent in the rest of the dataset. The challenge in the dataset used in Fier et al is the highly turbid environment. For Villon et al's method, misclassification of the dataset's textured background as a fish is reported to be the main reason for lowering their F-score, which totals up to 0.63.

There is still room for improvement for the proposed frameworks in this paper. In extended-UFFMS, there are faulty regions generated by the detection module that got passed on to the tracking module. In the Faster R-CNN method, insufficient training data, especially in low-light environments, caused a degradation in performance. Therefore, future work could aim to improve detection results and simulate or collect more data of fish in different environments to train a better classifier for the Faster R-CNN architecture.

## References

[1] Liang, C., & Pauly, D. (2017). Fisheries impacts on Chinas coastal ecosystems: Unmasking a pervasive ‘fishing down’ effect. *Plos One*, 12(3). doi: 10.1371/journal.pone.0173296.

[2] Appendix 2: Aquaculture regulatory framework. (n.d.). Retrieved from `https://www.mfe.govt.nz/publications/marine/aquaculture-risk-management-options/appendix-2-aquaculture-regulatory-\ \framework`

[3] A. Lukežič, T. Vojíř, L. C. Zajc, J. Matas, and M. Kristan, “Discriminative Correlation Filter with Channel and Spatial Reliability,” *International Journal of Computer Vision*, vol. 126, Issue 7, pp 671-688, July 2018.

[4] Tan, C. S., Lau, P. Y., Correia, P. L., & Campos, A., “Automatic analysis of deep-water remotely operated vehicle footage for estimation of Norway lobster abundance,” *Frontiers of Information Technology & Electronic Engineering*, 19(8), pp. 1042–1055, doi: 10.1631/fitee.1700720, 2018.

[5] Tan, C. S., Lau, P. Y., Correia, P. L., & Fonseca, P., “A Tracking Scheme for Norway Lobster and Burrow Abundance Estimation in Underwater Video Sequences,” *Joint Conference of IWAIT and IFMIA*, 2015.

[6] Fier, R., Albu, A. B., & Hoeberechts, M., “Automatic fish counting system for noisy deep-sea videos,” *2014 Oceans - St. Johns*, doi: 10.1109/oceans.2014.7003118, 2014.

[7] S. Villon, M. Chaumont, G. Subsol, S. Villéger, T. Claverie, and D. Mouillot, “Coral Reef Fish Detection and Recognition in Underwater Videos by Supervised Machine Learning: Comparison Between Deep Learning and HOG SVM Methods,” *Advanced Concepts for Intelligent Vision Systems Lecture Notes in Computer Science*, pp. 160–171, 2016.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1-9, 2015.

[9] . Jäger, E. Rodner, J. Denzler, V. Wolff, and K. Fricke-Neuderth, “SeaCLEF 2016: Object Proposal Classification for Fish Detection in Underwater Videos,” *CLEF*, 2016.

[10] R. Mandal, R. M. Connolly, T. A. Schlacher, and B. Stantic, “Assessing fish abundance from underwater video using deep neural networks,” *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018.

[11] Y. J. Ling and P. Y. Lau, “Fish monitoring in complex environment,” *Proc. SPIE 11049, International Workshop on Advanced Image Technology (IWAIT)*, 2019.

[12] “Grey Reef Sharks in Backlight Stock Footage Video (100% Royalty-free) 9555737,” Shutterstock. [Online]. Available: `https://www.shutterstock.com/video/clip-9555737-grey-reef-sharks-`. [Accessed: 13-Mar-2020].

[13] “A Large Group of Fish Stock Footage Video (100% Royalty-free) 27605698,” Shutterstock. [Online]. Available: `https://www.shutterstock.com/video/clip-27605698-large-group-fish-swim-red\ \-sea-egypt`. [Accessed: 13-Mar-2020].

[14] “School Fish Blue Background - Stock Footage Video (100% Royalty-free) 17734258,” Shutterstock. [Online].

[15] Hays, J. (2012, January). BLUEFIN TUNA FISH FARMING. Retrieved March 21, 2020, from `http://factsanddetails.com/world/cat53/sub340/item2188.html`

[16] Sustainable development of agriculture and fisheries industries. (2013, December 2). Retrieved March 21, 2020, from `https://www.ceo.gov.hk/archive/2017/eng/blog/blog20131118.html`

[17] R. Haddad and A. Akansu, “A class of fast Gaussian binomial filters for speech and image processing,” *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 723–727, 1991.

[18] J.Canny, “A Computational Approach to Edge Detection,” *Readings in Computer Vision*, pp.184–203. doi: 10.1016/b978-0-08-051581-6.50024-6, 1987.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[20] Koen E. A. Van De Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, “Segmentation as selective search for object recognition,” *2011 International Conference on Computer Vision*, pp.1879-1886, 2011.

[21] R. Girshick, “Fast R-CNN,” *2015 IEEE International Conference on Computer Vision (ICCV)*, pp.1440-1448, 2015.

[22] Y. Ren, C. Zhu, and S. Xiao, “Object Detection Based on Fast/Faster RCNN Employing Fully Convolutional Architectures,” *Mathematical Problems in Engineering*, vol. 2018, Article ID 3598316, 7 pages, 2018

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.779-788, 2016.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," Computer Vision – ECCV 2016 Lecture Notes in Computer Science, pp. 21–37, 2016.

[25] Gao, H. (2017, October 5). Faster R-CNN Explained. Retrieved from `https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8`

[26] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Ferrari, V. (2020). The Open Images Dataset V4. International Journal of Computer Vision. doi: 10.1007/s11263-020-01316-z

[27] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/cvpr.2016.90

[28] D. E. Rumelhart,G. E. Hinton, & R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 323(6088), pp. 533–536, `http://doi:10.1038/323533a0`

**Ling Yi Jun** received her Bachelor of Computer Science from Universiti Tunku Abdul Rahman, Malaysia in 2019. Currently, she is pursuing a master's degree in computer science in the same university. Her research interest include computer vision and machine learning.

**Lau Phooi Yee** received her BCompSc. from Universiti Teknologi Malaysia, Malaysia in 1996, MCompSc. from Universiti Malaya, Malaysia in 2002, and Ph.D. in Engineering from Keio University, Japan in 2006. She is currently an Associate Professor at Universiti Tunku