

UNDERWATER SPECIES-CONSTRAINED FISH
DETECTION USING MULTI-FRAME IMAGE
INFORMATION

LING YI JUN

MASTER OF SCIENCE (COMPUTER SCIENCE)

FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN
JANUARY 2024

ABSTRACT

UNDERWATER SPECIES-CONSTRAINED FISH DETECTION USING MULTI-FRAME IMAGE INFORMATION

Ling Yi Jun

Underwater fish detection system has many use cases such as fish biodiversity monitoring, aiding fish farming management, and providing data for marine resource management. Computer vision has proved to be a suitable tool for this fish detection task, as it is a low-cost, reliable, and most importantly, non-intrusive method for fish detection compared to trawling and other damaging methods. Detecting underwater objects introduces additional challenges, especially in unconstrained environments. Deep learning method has proved to be a powerful machine vision technique due to its deep hierarchical structures. YOLOv5 is used as an initial detector due to 1) its K-means clustering to select anchor box size and 2) its PANet detection head. Despite its strength, the result solely on YOLOv5 can still be improved, especially in decreasing the number of False Negative (FN).

We observed a research gap to improve detection performance when there are domain differences between training and testing data. We propose a method that aims to fill that gap. The proposed method integrates an auxiliary system with the original YOLOv5, which is successful in decreasing the number of FN, albeit introducing some False Positives (FP). The overall F1 score has

improved by 5.28%. This auxiliary system provides information to select low-confidence bounding boxes produced by YOLOv5, and thus it produces additional candidates (bounding boxes) for reducing FN probability. The first step in the auxiliary system is the Trail Image Formulation module, which constructs trail images that are domain-agnostic. A trail image contains the information of several image frames, which is derived from the concept of Motion History Images (MHI). Next, the detector of the auxiliary system is a modification of YOLOv5, and we name it YOLO-Ang. It takes in a trail image and produces bounding box candidates for each object in every frame. YOLO-Ang also produces angle information associated with the aforementioned bounding boxes. The output from YOLO-Ang is then processed using a Clustering-module and a simple Fusion module. To produce the final bounding boxes. In our extensive experiments, we compared three types of trail images (MHI), two types of YOLO-Ang, and two types of Clustering modules. The best version of the above variants is able to achieve over a 5% F1 score improvement.

ACKNOWLEDGEMENTS

I would like to express huge appreciation towards my thesis advisors, Prof. Hsueh-Ming Hang and Prof. Ching-Chun Huang, Prof. Kar-Hang Leung, and Ts. Siew-Cheng Lai for their continuous guidance and detailed feedback so that I can successfully finish the thesis.

Thank you to my family members as well for their encouragement and support. Even when things get tough, their support and care never wavered.

Lastly, thank you to my friends, both near and far, for their companionship and advice.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 15th January 2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Ling Yi Jun (ID No: 1905799) has completed this final year project/ dissertation/ thesis* entitled “ Underwater Species-Constrained Fish Detection using Multi-Frame Image Information ” under the supervision of Prof. Leung Kar Hang (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology , and Ts. Lai Siew Cheng (Co-Supervisor) from the Department of Computer Science , Faculty of Information and Communication Technology, and Prof. Hsueh-Ming Hang (External supervisor), from the Department of Electronics Engineering, and Prof. Ching-Chun Huang (External supervisor), from the Department of Computer Science.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

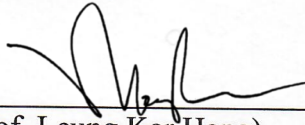
Yours truly,

Ling Yi Jun
(*Student Name*)

APPROVAL SHEET

This dissertation/thesis entitled “Underwater Species-Constrained Fish Detection using Multi-Frame Image Information” was prepared by LING YI JUN and submitted as partial fulfillment of the requirements for the degree of Master of Science (Computer Science) at Universiti Tunku Abdul Rahman.

Approved by:



(Prof. Leung Kar Hang)

Date:.....
Professor/Supervisor
Department of Computer Science
Faculty of Information and
Communication Technology
Universiti Tunku Abdul Rahman



(Prof. Hsueh-Ming Hang)

Date: Jan. 12, 2024
Professor/Co-supervisor
Department of Electronics
Engineering
National Yang Ming Chiao Tung
University



(Ts. Lai Siew Cheng)

Date: 11/1/2024
Professor/Co-supervisor
Department of Computer Science
Faculty of Information and
Communication Technology
Universiti Tunku Abdul Rahman



(Prof. Ching-Chun Huang)

Date:.....
Professor/Co-supervisor
Department of Computer Science
National Yang Ming Chiao Tung
University

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENTS	iv
SUBMISSION SHEET	v
APPROVAL SHEET	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xiv
CHAPTER	
1.0 INTRODUCTION	1
1.1 Motivation and Problem Statement	1
1.2 Project Scope	4
1.3 Objectives	5
1.4 Contributions	6
1.5 Report Organization	7
2.0 LITERATURE REVIEW	8
2.1 General object detection	8
2.2 YOLOv5	16
2.3 Fish Detection	20
2.3.1 Traditional machine learning methods	21
2.3.2 DNN-based single frame object detection	22
2.3.3 DNN-based multiple frame object detection	25
2.3.4 Summary	27
2.4 Motion History Images	33
2.4.1 MHI basics	33
2.4.2 MHI and DNN	40
3.0 PROPOSED METHOD	47
3.1 Trail Image Formulation Module	48
3.1.1 Trail image construction	51
3.1.2 Ground truth for trail image	56
3.2 YOLOv5 with Angle	58
3.3 Post-processing Module	61
3.3.1 Clustering module	61
3.3.2 Fusion module	70
3.3.3 Default parameters	72

4.0	ABLATION STUDY AND EVALUATIONS	74
	4.1 Fish Dataset	75
	4.2 Experimental Setup	78
	4.3 YOLOv5 Baseline Experiment	79
	4.4 Parameter Optimization	81
	4.5 Trail Image Formulation and YOLO-Ang Configuration	85
	4.6 Omitting Angle Information in YOLO-Ang and Clustering	90
	4.7 Replacing Trail Formulation by MHI	98
	4.8 Computational Complexity	103
5.0	CONCLUSION	107
	BIBLIOGRAPHY	109
	APPENDIX	121

LIST OF TABLES

Table		Page
2.1	Main improvements for each version of YOLO	16
2.2	Summary of selected works where enough information is published to be summarize	29
2.3	Sample dataset images	31
2.4	Results of the proposed method [80]	43
4.1	Parameter values used for clustering and fusion modules	82
4.2	Optimized parameter performance: trail image with addition method, trained with angle data, cluster module with angle data	84
4.3	Summary of best-performing models for different trail formulation methods	90
4.4	Summary of the differences between trail image formulation and general MHI	99
4.5	Summary of best-performing models of different trail formulation methods and their corresponding configurations	101
4.6	Selected works for comparison	102
4.7	Computational complexity of different variations for trail formulation methods	103
4.8	Computational complexity of YOLOv5 and proposed variants	104
4.9	Computational complexity of different clustering and fusion methods	104

LIST OF FIGURES

Figures		Page
1.1	Sample frames highlighting issues of underwater environments	3
2.1	The architecture of AlexNet [19]	9
2.2	The architecture of R-CNN [21]	11
2.3	The architecture of Fast R-CNN [22]	12
2.4	An illustration of Faster R-CNN architecture [23]	12
2.5	Illustration of SSD architecture [24]	13
2.6	Four composite styles for Dual-Backbone architecture [29]	14
2.7	The architecture of CARAFE [30]	15
2.8	Left: Training box size and ratio distribution for VOC2007; Right: Training box size and ratio distribution for F4K dataset	18
2.9	Left: Feature Pyramid Network (FPN); Right: Path Aggregation Network (PANet) [31]	18
2.10	Flowchart of the proposed system by Liu et al. [33]	22
2.11	Overall network structure of Composite FishNet [46]	25
2.12	Flowchart for a fish detection algorithm using motion information concepts [65]	25
2.13	Flowchart for another fish detection algorithm using motion information concepts [64]	26
2.14	Frame by frame MHI development [71]	34
2.15	Effect of different values of τ [71]	34
2.16	Actions with different δ values	34
2.17	Effects of unsuitable threshold value	35

2.18	Four different channels of optical flow are used to generate directional MHI [75]	36
2.19	Motion overwrite issue for sitting down and standing up motion (self-occlusion) [71]	36
2.20	HMHH Algorithm, extracted from [78]	39
2.21	Left image: MHI of a handwaving action. Right image: $D(:, :, :)$ of the red line of the left image.	39
2.22	Samples of an HMHH	40
2.23	MHI is computed using an OpenCV library, and DNN is used to produce action labels [79]	40
2.24	Flowchart of the creation of an RGB-MHI [80]	42
2.25	Proposed methods [80] - a) I3D with RGB-MHI attention b) I3D + RGB-MHI fusion	43
2.26	Proposed methodology [81]	45
3.1	System flowchart of our proposed method	47
3.2	Trail image subjectively highlights the movement of fairly still fishes better than frame difference	48
3.3	YOLOv5 gives low confidence to flipping fish	49
3.4	Top: YOLOv5 inability to assign high confidence score to camouflaged fish. Bottom: Boxes from YOLO-Ang output act as a complementary system to support low-conf YOLOv5 output	50
3.5	Trail image construction flowchart	51
3.6	Steps to encode time information for each frame difference, before added together	54
3.7	Illustration of determining the same fish instance from the previous frame	57
3.8	Illustration of a single layer of the detection head	58
3.9	Architecture of YOLOv5 [83]	59

3.10	The clustering module is responsible for processing the YOLO-Ang output to generate support bounding boxes, which are then used by the fusion module.	62
3.11	Samples of the YOLO-Ang outputs. Multiple fish objects are shown in this image. Four steps are designed to generate the final support boxes.	62
3.12	Another sample of YOLO-Ang output. An instance of a fish object is shown to illustrate the operation of Step a.4. The original RGB frame is also shown here for reference.	63
3.13	Sample of how the final supporting box is chosen in the case where angle data is omitted in clustering module	70
3.14	Fusion module flowchart: Low-confidence output that overlaps with the support boxes (satisfying certain criteria) is considered successful detection that eliminates FN errors.	71
3.15	More detailed flowchart for fusion module	71
3.16	Successful cases picked up by the supporting boxes. Left row: ground truth. Right row: pink boxes represent high-confidence detections; yellow boxes represent final support boxes	73
4.1	Six typical challenges that the FishCLEF-2015 dataset shows, extracted from [46]	77
4.2	Learning curves for training data	79
4.3	Learning curves for validation data	80
4.4	Learning curves on training data for model 4f4bb	86
4.5	Learning curve on validation data for model 4f4bb	86
4.6	Learning curves on training data for model 5f5bb	86
4.7	Learning curves on validation data for model 5f5bb	87
4.8	Learning curves on training data for model 6f6bb	87
4.9	Learning curves on validation data for model 6f6bb	87

4.10	Comparison between the addition and overlap methods	88
4.11	Comparison between different angle settings	92
4.12	Results of different models under default parameters for the addition method	94
4.13	Results of different models under default parameters for the overlap method	94
4.14	Difference in output between with and without training with angle data for trail images	96
4.15	Relationship between different angle settings and the average number of support boxes produced	97
4.16	Flowchart and visualization of constructing an MHI image	100
4.17	Results of different models under default parameters for the MHI method	101
4.18	Sample cases where an auxiliary system helps with detecting previously undetected fish.	106

LIST OF ABBREVIATIONS

MHI – Motion History Images

RGB - Red Green Blue

HSV – Hue Saturation Value

DNN - Deep Neural Network

HMHH - Hierarchical Motion History Histogram

YOLO – You Only Look Once

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION AND PROBLEM STATEMENT

Underwater fish detection system has many use cases such as fish biodiversity monitoring, aid in commercial fish farming management, and provide important data for marine resource management [1]. According to marine fishery resource surveys, fish detection proves to be a challenging research problem [2][3]. Therefore, it is important to develop or improve algorithms related to underwater fish detection. Thus, in recent years, it has become a popular research topic.

CNN and combination models are often used for fish detection [4]. YOLOv3, SVM, Faster R-CNN, and BM are the models that were often used and achieved good results. However, it is noted that the different algorithms proposed by the reviewed papers are made to solve problems from different data sets (such as [5][6][7]), and these different data sets all present different problems.

The purpose of fish detection is to separate a desired object from its background [8]. Over the years, camera equipment and technology have become more accessible, coupled with wide active research in the computer vision field; this enabled computer vision to be a suitable tool for this fish detection task, as

it is a low-cost, reliable, and most importantly, a non-intrusive method for fish detection compared to trawling and other damaging methods. Traditional techniques for computer vision focus on manually designed low-level features (also known as handcrafted techniques) such as colour, texture, contour, and shapes [3][9]. This introduces a generalization problem as it is tedious to construct all the features responsible for detecting different types of fish in different conditions. Moreover, the studies mentioned above that use handcrafted techniques are conducted in restricted conditions (controlled imaging environment with a calibrated stereo camera) that do not reflect real-world environments.

Detecting underwater objects introduces additional challenges, especially in an unconstrained environment where the backgrounds vary from site to site and even during different times of the day or seasons. Some and not all of the challenges for fish detection include: 1) unpredictable fish poses; 2) colour cast and low contrast; 3) blurry images due to sedimentation; 4) rapid changes in background due to light diffraction or moving aquatic plants; and 5) similarities between fish and background objects in terms of colours and shape. Some samples are shown in Figure 1.1.

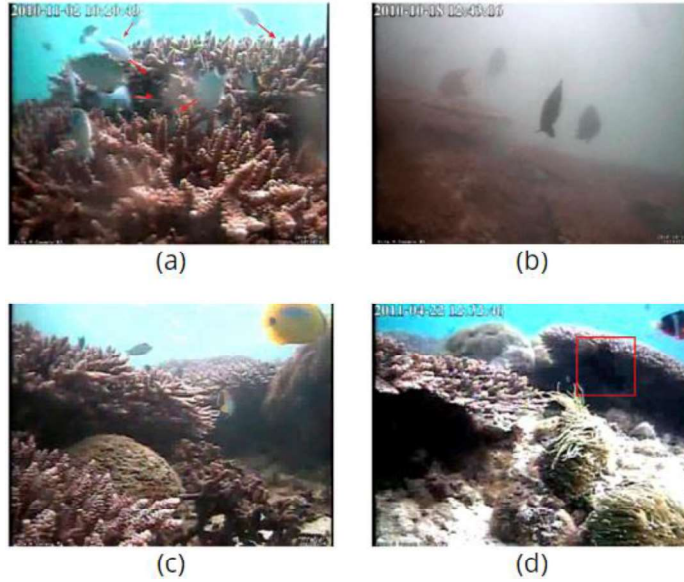


Figure 1.1: Sample frames highlighting issues of underwater environments. a) unpredictable fish poses. b) colour cast and low contrast, blurry images due to sedimentation. c) rapid changes in background due to light diffraction or moving aquatic plants. d) Similarities between fish and background objects in terms of colours and shape

In recent years, deep learning method has proved to be a powerful computer vision technique due to its deep hierarchical structures, which enable learning important features from training data and, thus, better generalization capabilities. However, due to its reliance on training data, when there is a difference in training and testing domains, there might be a drop in performance. Recent works in unconstrained underwater fish detection mainly adopt a complementary system approach, which is then integrated with the main DNN framework to improve performance. However, there are some research gap which leads to our problem statements:

1) There is little research work done to tackle the problem of domain differences between training and testing data. It is difficult to prepare a training set that covers every single underwater scenario (for example, some videos have a complex coral background with low visibility due to sedimentation, while another video will have a completely different background without corals but

with light refraction as noise). Therefore, this research aims to devise a method to address the domain gap.

2) Most existing methods depend on the visual information of a single frame. Due to the inconsistent visual conditions underwater, sometimes it is difficult for a deep learning model to extract relevant features for robust detection. Therefore, this research aims to leverage multi-frame information to enhance deep learning model performance, so that the model has increased opportunities to extract relevant info.

Our proposed method was able to create a data representation that is domain-agnostic. Our proposed method's modified DNN also makes use of features from multiple sequential frames. Then, the non-temporal DNN-based model is integrated with the temporal modified DNN model. General object detection deep learning models learn visual information; integrating a complementary system that learns another type of information can improve detection accuracy.

A more detailed analysis of recent works will be described in Chapter 2.0, the Literature Review section.

1.2 PROJECT SCOPE

This project explores and designs a complementary system that improves the *species-specific* fish detection accuracy of a deep learning model (YOLOv5). The system mainly aims to detect fish that fall within a specific set

of fish species that is defined in the training ground truth set without classifying each species. The dataset used in this project is called FishCLEF-2015 [10]. It is the dataset used for the fish identification task in the LifeCLEF-2015 competition. FishCLEF-2015 is a subset extracted from the data collected by the Fish4Knowledge project [11], which records and analyses 90 thousand hours of video from camera locations on several tropical coral reefs off the coast of Taiwan.

1.3 OBJECTIVES

As mentioned, in the field of computer vision, deep learning is one of the more popular and accurate methods for object detection purposes. However, deep learning has its own technical problems and limitations. Therefore, this work aims to explore those limitations and solutions to improve detection accuracy:

1) In order to solve the first problem statement (domain differences between training and testing data), our objective is to generate training data representations that is domain agnostic. Which is helpful to extract different types of information that are not normally detectable in the original training data, such as the direction the object is facing and the movement of the object. The intuition is that this can help the model to learn better and achieve a higher accuracy.

2) In order to solve the second problem statement (dependency on visual information in a single frame), our objective is to integrate temporal (made out of multiple sequential frames) with non-temporal deep learning models, to make

use of both types of information – spatial and temporal, thus, the proposed method does not rely on information of only a single frame. This can help the model use information from other frames thus decreasing the model’s False Negative rate.

1.4 CONTRIBUTIONS

We modified the YOLOv5 [12] architecture to learn the motion-based features from a specific data representation called trail. Its output is then merged with spatial features learned from the original YOLOv5 architecture to take advantage of using both spatial and temporal information. The main contributions of this project include:

- 1) Pre-processing data to include temporal information: The fish “trail” is formed by overlapping or adding the frame differences of multiple frames. Because the camera is stationary, the frame differences contain mostly moving objects. Then, the trail images are used in the subsequent modules for further processing. This step is particularly useful to differentiate between fish and underwater non-fish objects.
- 2) Modify the YOLOv5 architecture to generate the angle information, “YOLO with angle.”: This is useful to predict the direction the fish is moving.
- 3) Fuse the temporal and spatial information by combining the YOLOv5 outputs and the outputs from our modified YOLOv5 operating on the trail images. The first step in this processing is a “Clustering module”

which condenses the outputs of “YOLO with angle” into a few selected candidates. Then, the low-confidence outputs are merged with the Cluster module outputs in the “Fusion module” to generate detected objects in addition to the original YOLOv5.

1.5 REPORT ORGANIZATION

The rest of the chapters are organized as follows: Chapter 2 will describe some project background and review previous work that the proposed system is built upon; Chapter 3 details the proposed system; Chapter 4 includes experiment results and ablation studies; and lastly, Chapter 5 concludes the work and discusses future work.

CHAPTER 2

LITERATURE REVIEW

2.1 GENERAL OBJECT DETECTION

The purpose of object detection is to detect instances of visual objects in digital images. Currently, object detection can be classified into two categories: traditional handcrafted method and the deep learning method.

Before deep learning gained popularity, in order to localize objects in a single image, the traditional object detection scheme usually used a sliding window to locate the object. The possible location of the object is obtained through scanning the whole image, and then the object region is identified. Manually designed features are then extracted from this object region, and these features are then classified using techniques such as the support vector machine (SVM) [13]. Some feature extraction techniques include the General Hough transform [14] for geometric feature extraction. Another technique is the Harris corner detector [15] which extracts object features by detecting corners from two images and calculating the degree of correlation between them to detect objects. The above 2 methods are sensitive to changes in image size, rotation, and gray value. A feature extraction technique that is robust against the problems mentioned earlier is SIFT (Scale-Invariant Feature Transform) [16]. The SIFT algorithm detects and describes local features of an image. It treats the feature itself as an object, and thus, the image's rotation and scale do not affect the result.

Another feature that can be used is the Haar-like feature, which is similar to the Haar wavelet. The Haar-like features were used in the first real-time face detector [17]. Another popular feature is the histogram of oriented gradients (HOG), which uses the concept that local object shape and appearance in an image can be described by the distribution of edge directions or intensity gradients. An image is divided into cells, and these intensity gradients are compiled into a histogram of gradient directions for each pixel within each cell of the image [18].

Deep learning techniques gained popularity in 2012 when the AlexNet [19] architecture was introduced, which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. The architecture achieved a top-5 error rate of 15.3%, a huge leap from the runner-up which had a top-5 error rate of 26.2%. The architecture is made up of 5 convolutional layers. The 1st, 2nd, and 5th layers have a max pooling layer for proper feature extraction. The 6th and 7th layers are fully connected layers, which are followed by a SoftMax layer. Figure 2.1 shows the structure of the architecture.

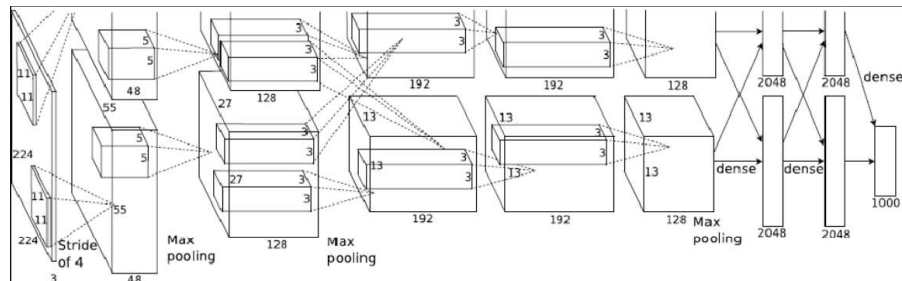


Figure 2.1: The architecture of AlexNet[19]. The architecture is split into 2 branches to run 2 GPUs in parallel. The input image is subsampled to a 224 x 224 pixel tensor with a stride of 11 x 11 pixels.

After AlexNet was proposed, there have been many variants of deep learning image classification architecture. Initially, for object localization in the object detection task, an inefficient and computationally expensive sliding window approach was used, and then object classification was done in each location. Nowadays, the DNN-based object detection architectures can be further split into 2 categories: Two-stage and one-stage detectors.

The two-stage detector's architecture contains 2 stages. The first stage mainly uses region proposals technique to generate region of interests (RoIs), and then these RoIs are sent through the pipeline to the second stage, object classification and bounding-box regression. A widely used two-stage object detection architecture is the Region-based Convolutional Neural Network (R-CNN) family [20]. Single-stage detectors treat object detection as a simple regression problem. Usually, single-stage detectors have a lower accuracy rate but are much faster than two-stage object detectors. The popular single-stage detectors are the SSD (Single Shot MultiBox Detector) and the YOLO family.

The architecture of R-CNN [21] can be summarized in Figure 2.2. The first stage of R-CNN uses selective search to extract approximately 2000 candidate region proposals. These region proposals are then warped to a fixed size. Then, these warped region proposals are propagated through a CNN to generate a feature vector. The feature vector is then classified with a trained binary SVM. The binary SVM was independently trained for each class. Using CNN features, a trained regression model is used to correct the predicted detection window to reduce localization error.

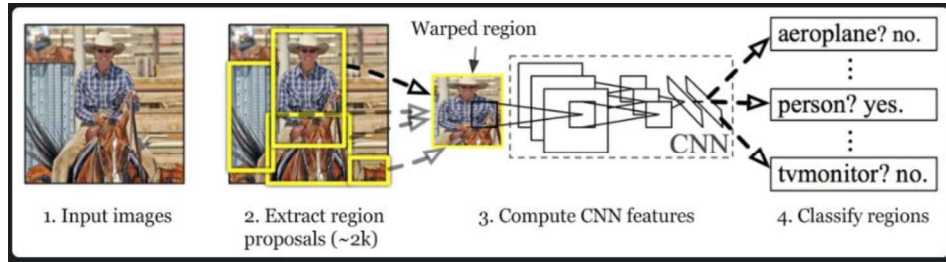


Figure 2.2: The architecture of R-CNN [21].

The R-CNN architecture has a bottleneck, which is the first stage where selective search is used to extract approximately 2000 candidate region proposals. Then, for every single region proposal, CNN feature vectors need to be generated. This process involves 3 models that do not share any computation. Therefore, Fast R-CNN [22] is proposed to make computation more efficient. Instead of extracting CNN features for each region proposal independently, the entire image is propagated through a CNN to get a feature vector. Then, a selective search will be performed on that feature matrix, extracting 2000 region proposals. The same feature matrix is used for both learning the object classifier and bounding-box regressor. The sharing of computation speeds up training and inference in the previous version of R-CNN. However, the region proposals are still extracted by another model, which Faster R-CNN [23] is able to further speed up.

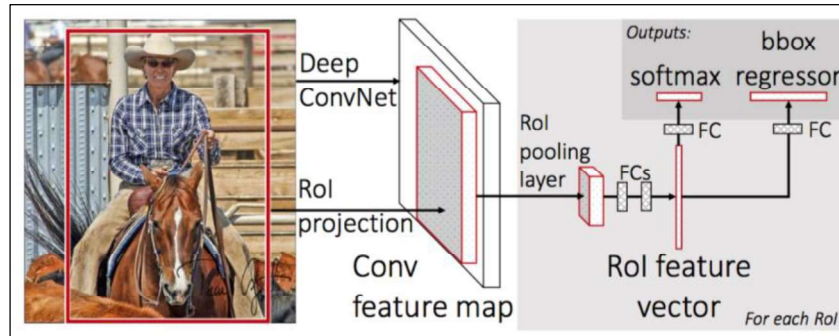


Figure 2.3: The architecture of Fast R-CNN [22].

Faster R-CNN further streamlines efficiency by integrating the region proposal algorithm into the CNN model. The region proposal network (RPN) shares a convolutional feature layer with Fast R-CNN to make Faster R-CNN. The RPN and CNN can be trained end-to-end.

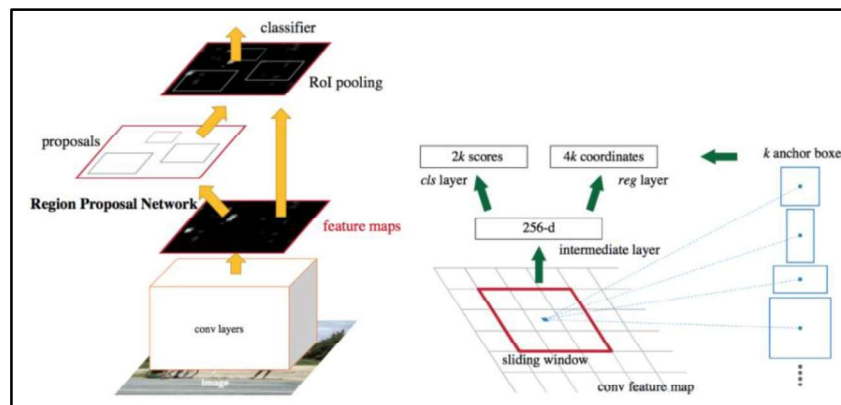


Figure 2.4: An illustration of Faster R-CNN architecture [23].

Single Shot MultiBox Detector (SSD) [24] is a one-stage object detector. It uses an anchor mechanism and an end-to-end, one-step structure where object classification and location regression happen directly in the convolution stage. SSD uses the VGG-16 network as the backbone, but the last two fully connected

layers are replaced with convolutional layers, and another four convolutional layers are also added to form the feature extraction network.

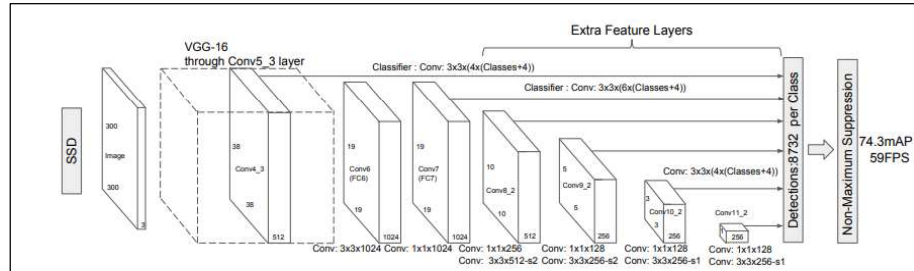


Figure 2.5: Illustration of SSD architecture. Several convolutional layers are added to the end of the base network [24].

The You Only Look Once (YOLO) family is also a family of one-stage detectors. The first YOLO model [25] was introduced to perform object detection in just a single stage, thus increasing inference speed. The model divides an image into grid cells. Then, for all the grid cells, the probability of an object residing in that grid cell is calculated. The algorithm then groups nearby high-probability cells as a single object. YOLOv2 [26] is an improvement over YOLOv1, as it is capable of detecting 9000 categories of objects. Some of the technical improvements include introducing anchor boxes. Anchor boxes are predefined areas for an image. Instead of setting the size and aspect ratio of the anchor box randomly, the dimensions of clusters are preselected to make sure that the anchor boxes fit the training data best. This seems to help enhance the accuracy. YOLOv2 also goes through random resizing throughout the training process (multi-scale training). YOLOv3 [27] further improves YOLOv2 by not using fully connected or pooling layers, which reduces the model size and number of weights. Instead, it uses residual models for multiple feature learning

with a feature pyramid network. YOLOv4 [28] further improves YOLOv3 with technical improvements that can be categorized into 2 parts: a bag of freebies (techniques that enhance model performance without increasing computational cost) and a bag of specials (techniques that increase accuracy but also increase computation cost). The bag of freebies includes techniques like mosaic data augmentation, bounding box regression loss, different types of regularization, and normalization. The bag of specials includes skip connections like cross-stage partial connections, spatial attention modules, and non-linear activation functions.

Other DNN models that are based on some standard DNN models include: Cbnet + Dual_ResNet50 [29], which use a novel strategy where multiple identical backbone networks are assembled with composite connections between the adjacent backbones. Composite connections are made up of a $1 \times q$ convolutional layer and batch normalization layer; this is to reduce the number of channels. Then, there is an up-sample operation. There are multiple versions of composite style, as shown in Figure 2.6. AHCL seems to perform the best out of the versions.

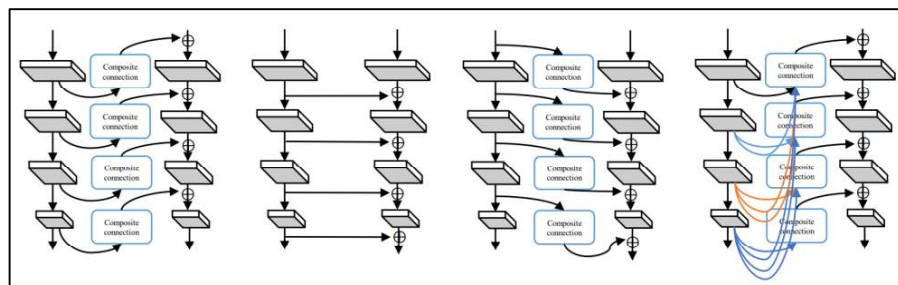


Figure 2.6: Four composite styles for Dual-Backbone architecture [29]. From left to right: Adjacent Higher-Level Composition (AHCL), Same Level Composition (SLC), Adjacent Lower-Level Composition (ALLC) and Dense Higher-Level Composition (DHLC)

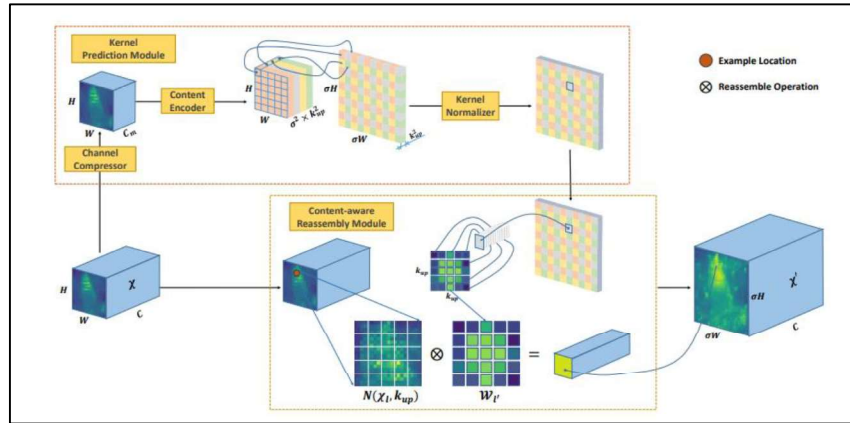


Figure 2.7: The architecture of CARAFE [30]. CARAFE is made up of two components: 1) kernel prediction module, and 2) content-aware reassembly module.

The concept behind Cascade R-CNN + ResNeSt101 + Carafe [30], is that the up-sampling kernel usually only exploits the spatial position of a pixel, which overlooks the semantic information of the feature map. Therefore, CARAFE consists of two steps: a reassembly kernel prediction for each target location according to its content, and the second step, which is to reassemble features with predicted kernels. The first step has 3 sub-steps: 1) channel compressor- reduces model parameters by reducing the channel of the input feature map; 2) the content encoder – takes the compressed feature map and encodes its content to generate the reassembly kernel; and 3) kernel normalizer – which is to apply the Softmax function to each reassembly kernel.

2.2 YOLOV5

During the start of this project, YOLOv5 was the most popular and stable open-source code base on GitHub, over the years; YOLOv6 [34], YOLOv7 [35] and v8 [36] were published. Due to it being recently published, there are a lack of published papers that utilize YOLOv5,6,7 and 8 to make a formal comparison. However, initial reports from renowned forums, discussion

pages, and blogs claim better speed and accuracy improvements for each iteration of YOLO [37] [38] [39] [40] [41]. In terms of research with fish detection, there is little literature using YOLOv5. Most papers that use the YOLO family are mostly using YOLOv3, such as [42] [6] [33]. Papers such as [32] and [43] use YOLOv1. To the best of our knowledge, for the purpose of fish detection, there is very little literature that employs the YOLO version above v5. A quick analysis is done to see what changed between YOLOv5, 6, 7 and 8 according to multiple sources on the internet. The latest version, YOLOv8, was published by the same YOLOv5 GitHub authors. Table 2.1 shows the main improvements each version of YOLO made after YOLOv5, according to the published paper for each YOLO version.

Table 2.1: Main improvements for each version of YOLO

Model	Changes from the previous version
YOLOv6 [34]	<ul style="list-style-type: none"> • Different scales architecture varies, where small models use a plain single path backbone and larger models use efficient multi-branch blocks. • Self-distillation strategy is used with dynamically adjusted knowledge from the teacher and labels to help the student model learn knowledge more efficiently. • Quantization scheme is reformed with RepOptimizer and channel-wise-distillation for faster and more accurate detector.
YOLOv7 [35]	<ul style="list-style-type: none"> • Proposed compound scaling method: The depth of computational block is scaled up by 1.5 times, and the transition block's width is scaled up to 1.25 times. • Proposed planned re-parameterized model: The 1x1 and 3x3 convolutional layers in the dark block's position are reversed to fit their re-parameterized model design strategy. • Proposed assistant loss for auxiliary head: Before merging cardinality, the auxiliary head is connected after one of the sets of feature maps. This design allows the weights of the newly generated feature map to not be updated directly by assistant loss; thus, the pyramid

	of the lead head can get information from objects of different sizes.
YOLOv8 [36]	Authors of YOLOv8 have yet to write a paper about their findings, but they did a summary on their websites of the updates that are made for YOLOv8 as follows: a new backbone network, a new anchor-free split head, and new loss functions.

YOLOv5 has gained popularity in the past two years due to its accuracy and speed. Moreover, the open source easy to use GitHub repo [12] made it accessible to a large audience. The architecture's Path Aggregation Network [31], coupled with its anchor box's implementation, allows the YOLO family architectures to detect objects of various aspect ratio and sizes, with emphasis on small objects. Thus, this makes the YOLO family architectures suitable for fish detection especially in unconstrained environments, as can be seen in previous subchapter where work that uses YOLO [6] [32] [33] were mentioned. Therefore, our proposed system mainly evolves around the YOLOv5 architecture. Details on parts that make YOLOv5 a suitable choice for fish detection are as follows:

1) K-means clustering to select anchor box size and ratio:

Running k-means clustering on the dimensions of the training set's bounding boxes allows YOLO to start with better anchor priors without having to hand-pick them. One note regarding the k-mean's default distance metric, which is the Euclidean distance, is that it generates more error for larger boxes; therefore, a distance metric that reflects the objective, which is good IOU scores, looks like $d(box, centroid) = 1 - IOU(box, centroid)$. "Box" represents the ground truth; "centroid" represents anchor box priors. Good anchor box priors that represent the dataset improve the YOLO algorithm by 5% with the

VOC2007 dataset [44] compared with handpicked priors.

Below are samples of the differences in bounding box size and aspect ratio between the VOC2007 and F4K datasets. VOC2007 boxes favor thinner, taller boxes, whereas F4K's boxes are smaller and longer in length.

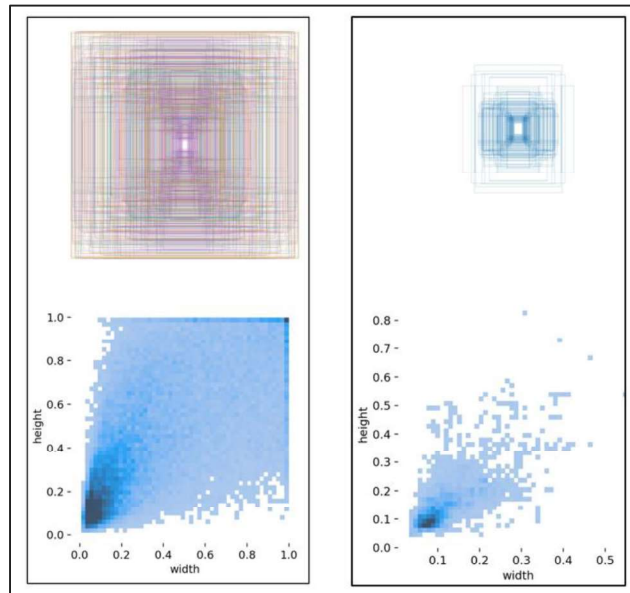


Figure 2.8: Left: Training box size and ratio distribution for VOC2007; Right: Training box size and ratio distribution for F4K dataset

2) PANet detection head [31]:

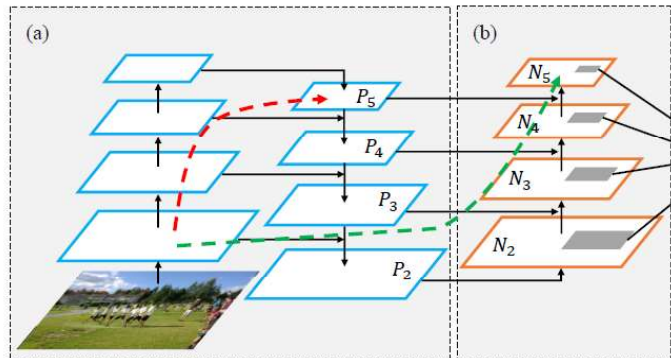


Figure 2.9: Left: Feature Pyramid Network (FPN); Right: Path Aggregation Network (PANet) [31].

What helps PANet localize small objects at high accuracy is its emphasis on both semantically and spatially strong feature map representation made possible by lateral connections. To begin, FPN's top-down pathway (the right branch of FPN) is able to attain higher resolution by up-sampling spatially coarser but semantically stronger feature maps from higher pyramid levels. The bottom-up feature map (the left branch of FPN) is of lower-level semantics, but its activations are more accurately localized as it was subsampled fewer times. Lateral connections combine the strengths of both spatial and semantic features by performing concatenation of layers from the left and right branches according to their level.

PANet is an extension of FPN by introducing another bottom-up structure. Low-level features are known for having a high response to low-level patterns (edges and instance parts), which is a strong indicator of accurately localized instances. Therefore, the authors of PANet implemented a clean lateral connection bridging low levels to the top levels. The "shortcut" (green dashed line in Figure 2.9) propagates strong responses of low-level patterns of FPN to the top levels of PANet. This enables PANet to produce semantically strong and spatially detailed feature maps.

Despite its strength, our preliminary experiments show a high number of False Negative. Our proposed system aims to improve that with concepts explained in Chapter 1.

2.3 FISH DETECTION

There are some research interests in image processing techniques for fish detection. For our review, the reviewed works are briefly categorized into traditional handcrafted methods; there are more to elaborate on in traditional machine learning methods; the more recent deep learning methods. For deep learning methods, it can be further categorized into single-frame, or multi-frame (including temporal information) object detection.

Some work has been done for traditional handcrafted fish detection. A review paper [45] summarizes the handcrafted methods into 3 categories: 1) Colour feature, which refers to the colour channel histogram features of an image; 2) textural feature, which denotes the feature extracted from an image, such as histogram of grey difference, and grayscale co-occurrence matrix; and 3) geometric features, such as shape contour feature, shape region features, and position feature of an object in an image. An example of a paper using colour feature is by [14], which employs the boundary effect between the background and foreground peak of the H-component of the HSV colour space. This technique is robust against different illumination conditions.

One drawback of traditional handcrafted methods is that image feature extraction relies on manually designed features. This requires relevant professional knowledge, and many different feature sets are required to cater to the many different underwater environments presented in offshore situations, which tend to be unfeasible. Machine learning and deep learning methods help

automate feature learning, making object detection features more robust.

2.3.1 TRADITIONAL MACHINE LEARNING METHODS

Before DNNs, there were some traditional machine learning methods that did not rely on convolutional neural networks. For methods that do not use temporal information, Tahnim et al [47] proposed using a Histogram of Oriented Gradient (HOG) descriptor with Support Vector Machines (SVM) to detect fish. Le and Xu [48] proposed an improved OTSU algorithm that combines class probability and grey level histogram interval. Yan and Xiang [49] optimized a kernel function and partitioned it to map the data set into a high-dimensional space with a kernel-based K-nearest neighbor algorithm.

One image property that can be exploited in the fish detection task is the assumption that fish rarely stay stationary for long periods of time; therefore, pixels of moving objects can be regarded as potential fish objects. Hsiao et al. [50] extracted motion information by using adaptive background subtraction. Gaussian Mixture Models (GMM) are used to model background pixels. Then, a fish category database is created. It is made up of various features from different fish species that are taken at various angles and illumination conditions. A sparse representation-based classification is then proposed for fish recognition and identification, with features extracted using Eigenfaces and Fisherfaces. Palazzo and Murabito [51] proposed a method where covariance modeling is used to model the background and foreground in the video frames using colour and scale texture features.

2.3.2 DNN-BASED SINGLE FRAME OBJECT DETECTION

There are some works that use DNN for single frame object detection, such as [6] and [32] which use YOLO directly. Liu et al. [33] use YOLO with a parallel correlation filter. The proposed algorithm first does image enhancement with a haze removal algorithm. Each channel of the image is stretched pixel by pixel using two thresholds that can be backpropagated. Then, YOLOv3 does the object detection. Lastly, a correlation filter-based tracker is used to improve speed and performance.

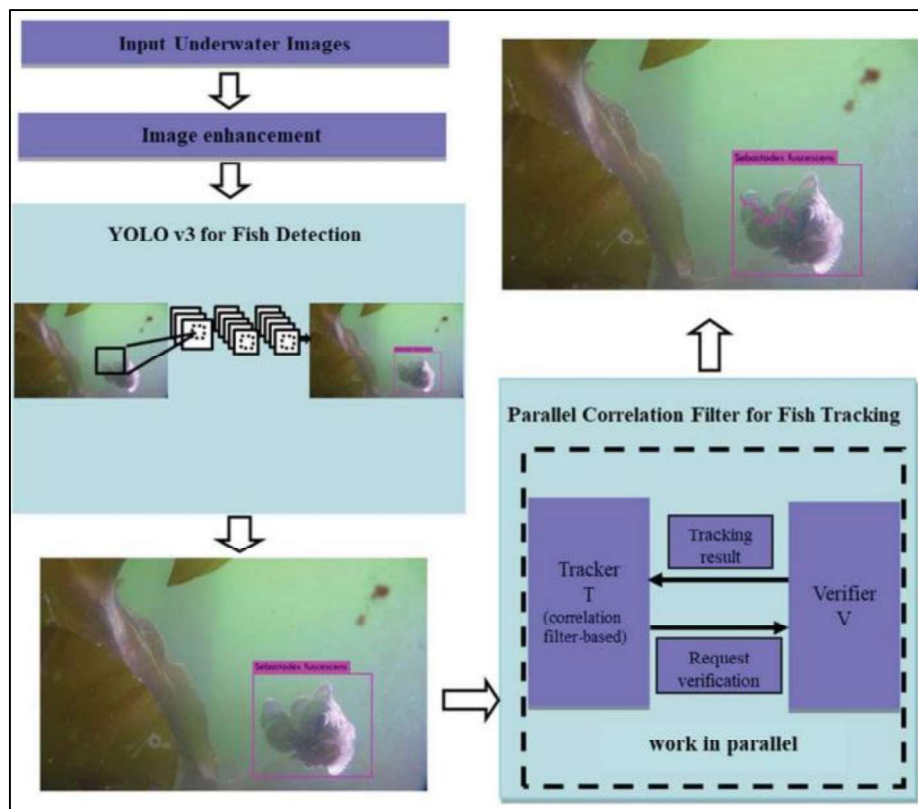


Figure 2.10: Flowchart of the proposed system by Liu et al [33].

Jäger et al. [43] use multi-class SVM to process features extracted from the AlexNet architecture to perform classification. The region proposals for classification are generated through background subtraction. Zhuang et al. [52] first use the SSD architecture to separate background and foreground objects. Then, another detection architecture, PVANET, is used to detect foreground fish, and a bounding box is generated. Lastly, a background image is computed by selecting the median value at each pixel position for every video; background subtraction and erosion are done to create a mask for each frame. If the previously generated bounding box's background area is larger than a threshold, then that bounding box is eliminated. The rest of the boxes are detected as fish.

Some efforts in improving deep learning models to detect fish include the method proposed by Li et al. [53], where they adopted a region proposal network from Faster R-CNN and shared it with the Zeiler and Fergus (ZF) model to produce better detection performance. Xue and Ju [54] optimized an AlexNet model by removing part of a redundant convolution layer and then combined it with a flexible attention algorithm. Abinaya et al. [55] proposed a naïve Bayesian fusion deep learning network layer. Shi et al. [56] proposed FDDet, which added a feature fusion module in SSD to enhance feature representation by aggregating adjacent prediction layers. Wu et al. [57] use stereo images since stereo means there's 2 frames at a time. The "left" image is to detect fish using SSD, and the "right" image's detected fish act as a complementary detection to the left one. To reduce the FP rate, a histogram of gradients is used as a feature for SVM to verify correct matches within an image pair. Li et al. [58] used Ghost convolution to replace the convolutions in

YOLOv5. A new attention mechanism is also added to the feature extraction network. Hu et al. [59] modified the connection between the FPN network and PANet and replaced feature mapping in the YOLOv4 architecture to be more fine-grained. Prasetyo et al. [60] fuse low-level and high-level features across the depth of the CNN using “depth-wise separable convolution” (DSC) to improve the performance of the VGG architecture. Zhang et al. [61] solve the noisy background problem by adding a new term in the loss function of the ResNet50 model; this term allows the network to discriminate between fish regions and noisy background, so the network pays more attention to the fish regions. Villon et al. [62] used the GoogLeNet architecture to extract fish features and the soft-max classification method to detect coral fishes; in order to optimize the model, the authors added a decision rule.

Zhao et al. [46] proposed a method called Composite FishNet. Frames without fish instances are used to train the source domain backbone, which is an auxiliary backbone to adaptively learn the scene information of each source domain. Thus, the interference of underwater environment information on object characteristics is reduced.

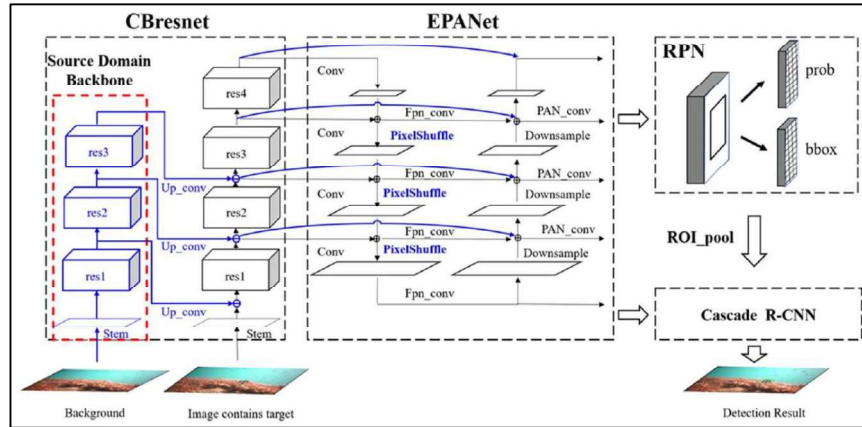


Figure 2.11: Overall network structure of Composite FishNet [46]

2.3.3 DNN-BASED MULTIPLE FRAME OBJECT DETECTION

There are a number of works that utilize motion information for the fish detection task. One way to extract motion information is to calculate optical flow. Examples are [63], [64], and [65]. Below is the flowchart for [65].

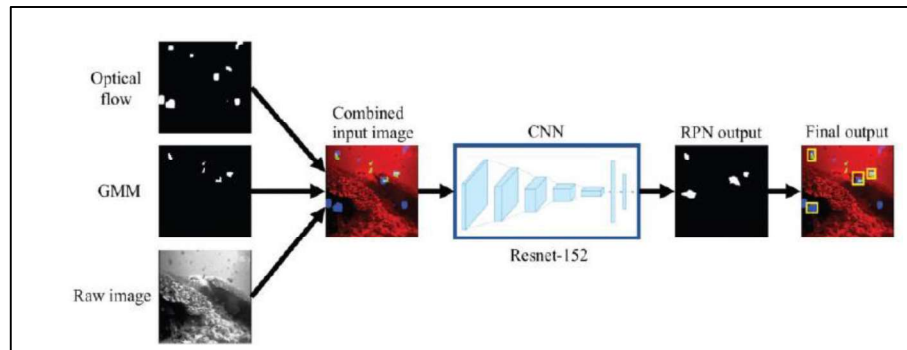


Figure 2.12: Flowchart for a fish detection algorithm using motion information concepts [65].

The authors of [65] proposed a hybrid system combining motion-based features, optical flow, and GMM (Gaussian Mixture Modeling), which are then further combined with a raw greyscale image and fed to the CNN for

training. The other algorithms [63] [64], with concepts following the same structure as Salman et al. [65], Jalal et al. [63] similarly combined optical flow and GMM, but it went through CNN processing before going through a combination algorithm with another CNN-processed raw RGB image.

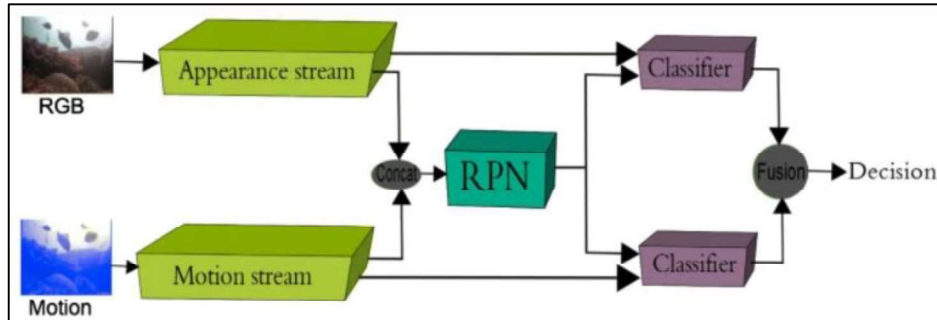


Figure 2.13: Flowchart for another fish detection algorithm using motion information concepts [64]

For Tamao et al. proposed method [64], motion images and raw RGB images have two points of combination: concatenation is performed before going through an RPN network, and another one after RPN and before a classifier; fusion is then performed to get the final result. However, one major drawback of using optical flow is the high computational complexity of its implementation.

Just a quick note: for works such as [63] [64] [65] [50] [51], where their common characteristic is using motion information to help with detection, a static background without moving objects is needed to model source domain information. In cases where backgrounds are not constant, or if there are other non-fish moving objects that act as noise, which is the case in an unconstrained underwater environment, there might be a need to manually pick out non-fish

frames for every domain change.

Shen and Nguyen [66] proposed a framework that is based on RetinaNet; n continuous frames go through ResNet to form $3 \times n$ feature maps. These feature maps then pass through a feature pyramid network to get pyramid features. Then, these features are combined to form a 3D feature, and then bounding box regression is performed by a 3D regression subnet. Labao and Naval [67] use Faster R-CNN with an LSTM model cascade structure to optimize detection accuracy.

2.3.4 SUMMARY

This paragraph will observe some of the works mentioned above and state some possible improvements to the methods described above. The authors of [6] and [32] solely use the YOLO architecture without other support systems.

When the appearance of a fish is dissimilar to the training set, which often happens due to the complex underwater environment, it will be falsely undetected. Liu et al. [33] use a correlation filter, which might fail for fast-moving fish, and it is difficult to correlate between frames where there is heavy sedimentation, as it could correlate noise instead of the fish itself. One of the solutions to this problem is to include information from multiple frames and have a CNN learn its correlation. Jäger et al. [43] extracted features from AlexNet that do not take different object sizes into consideration. YOLO's PANet skip connection helps to pull spatial information from earlier layers to

later layers. Zhuang's et al. [52] background modelling technique makes it difficult to detect fish under camouflaged conditions, especially for complicated backgrounds such as moving corals and illumination changes. Learning movement behaviour lowers the chances of fish being missed out due to camouflage. This is because it is easier to detect patterns in multiple frames. The method proposed by Liang et al. [29] greatly increases parameters since whole backbones are added. A problem with adding such backbones is that the features learned might be redundant, which wastes computational resources. One possible solution is to have the supporting model learn the features that are different from the original RGB frames (such as MHI).

According to the work mentioned above, most standalone DNN frameworks are insufficient for fish detection tasks. Usually, complementary systems or modifications are integrated with the main DNN framework to improve performance. In order to further increase robustness, temporal information should also be exploited. Our proposed method aims to do that.

Table 2.2: Summary of selected works where enough information is published to be summarize

Method	Standard baseline model	Dataset and description	Metric used; Accuracy(%)	Baseline Accuracy	Amount of improvement
Cascade Mask R-CNN + ResNeXt152 + Triple Backbone [29]	Cascade Mask R-CNN + ResNeXt-152	COCO test-dev [68] (non fish)	mAP@50; 69.80	67.00%	2.80%
Mask R-CNN + CARAFE [30]	Mask R-CNN + GUM	COCO test-dev [68] (non fish)	mAP@50; 56.20	55.70%	0.50%
YOLO (without changes) [6]	YOLO	Self-made <ul style="list-style-type: none"> High turbidity Part grayscale 	mAP@50; 53.92	-	-
Fish_AlexNet [54]	AlexNet	QUT_fish <ul style="list-style-type: none"> Variable light condition Controlled, steady condition White background 	Accuracy = $\frac{TP + TN}{TP + FP + TN + FN}$; 97.43	93.35%	4.08%
FFDet [56]	SSD	LCF-15 <ul style="list-style-type: none"> Complex background Blurry scenes Varying illumination 	mAP@50; 61.40	57.63%	3.77%
Ghost-YOLOv5 [58]	YOLOv5	Undefined <ul style="list-style-type: none"> Close ups Complex background Large number of fish 	mAP@50; 76.10	72.20%	3.90%
SSD + Stereo + SVM [49]	SSD	Self-made <ul style="list-style-type: none"> Stereo Fish tank scene 	mAP@50; 77.70	76.41%	1.29%



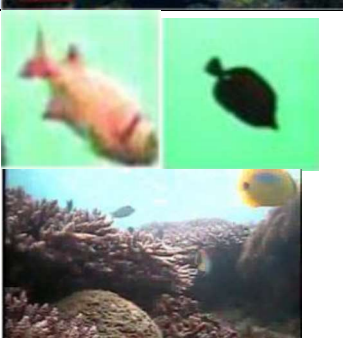
Table 2.2 continued

Method	Standard baseline model	Dataset and description	Metric used; Accuracy(%)	Baseline Accuracy	Amount of improvement
PVANET + SSD [52]	SSD	SeaCLEF 2017 <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	Counting score; 88.00	87.00%	1.00%
YOLO + GMM [63]	YOLO	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	F-score; 95.47	90.67%	4.80%
Faster R-CNN + Shared RPN fusion [64]	Faster R-CNN	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	mAP@50; 70.50 F-score; 80.22	67.49% 78.78%	3.01% 1.44%
Hybrid System [65]	R-CNN	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	F-score; 80.02	77.30%	2.72%
2RPN + LSTM [67]	Faster RCNN	Self-made <ul style="list-style-type: none"> • Varying illumination • Small sized fishes • Large number of fish (20-200) 	F-Score; 44.21	28.43%	15.78%
3D RetinaNet [66]	Retina-Net	Self-made <ul style="list-style-type: none"> • Large number of fish • Complex background 	mAP@50; 73.30	69.30%	4.00%

Table 2.3: Sample dataset images

Dataset name	Sample frames
Description	
Papers that use it	
Self-made	
<ul style="list-style-type: none"> • High turbidity • Part grayscale 	
[56]	
QUT-Fish	
<ul style="list-style-type: none"> • Variable light condition • Some in controlled, steady condition • Some have white background • Cropped 	
[54]	
Undefined	
<ul style="list-style-type: none"> • Close ups • Complex background • Large number of fish 	
[58]	
Self -made	
<ul style="list-style-type: none"> • Stereo • Fish tank scene 	
[49]	

Table 2.3 continued

Dataset name	Sample frames	
Description		
Papers that use it		
<p>Self-made</p> <ul style="list-style-type: none"> • Varying illumination • Small sized fishes • Large number of fish (20-200) 		
[67]		
<p>Self-made</p> <ul style="list-style-type: none"> • Large number of fish • Complex background 		
[66]		
<p>LCF-15 [with extra ROI images included]</p> <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 		
[56][52][63][64][65]		

2.4 MOTION HISTORY IMAGES

2.4.1 MHI BASICS

Motion History Images (MHI) is a concept where multiple sequential frames extracted from a video can be represented in a single image, encoding moving parts and motion flow into a single frame [69]. MHI is useful in low-illumination conditions due to its inherent algorithm [70]. Thus, it is suitable to implement in cases where illumination is not consistent, such as underwater videos. MHI are mostly used in human action recognition before the use of the CNN model. In the advent of convolutional neural networks, MHI is used as either a combination or a complement to CNN models.

The MHI $H_{\tau}(x, y, t)$, can be represented by the equation below.

$$H_{\tau}(x, y, t) = \begin{cases} \tau & \text{if } \Psi(x, y, t) = 1 \\ \max(0, H_{\tau}(x, y, t-1) - \delta) & \text{otherwise} \end{cases} \quad (2-1)$$

(x, y) and t represents position and time. $\Psi(x, y, t)$ is the update function; it also represents the moving pixels of an object in the current frame. τ represents the duration, which determines the temporal extent of the movement (frames), and δ is the decay parameter. The MHI definition results in a scalar-valued image, where brighter pixels are the more recently moving pixels and vice versa. Figure 2.14 [71] shows the frame-by-frame development of MHI frames. This shows that MHI records the temporal history of motion. Usually, MHI is made up of a binarized image obtained from background subtraction using a set threshold. Figure 2.15 [71] shows how τ affects the final

MHI. If the τ value is smaller than the number of frames used to construct the MHI frame, then prior motion information is lost. Figure 2.16 [71] shows the effect of the δ decay parameter on MHI. If there is no motion in a specific pixel when there used to be motion in the previous frame, the pixel value will be reduced by δ .



Figure 2.14: Frame by frame MHI development [71].

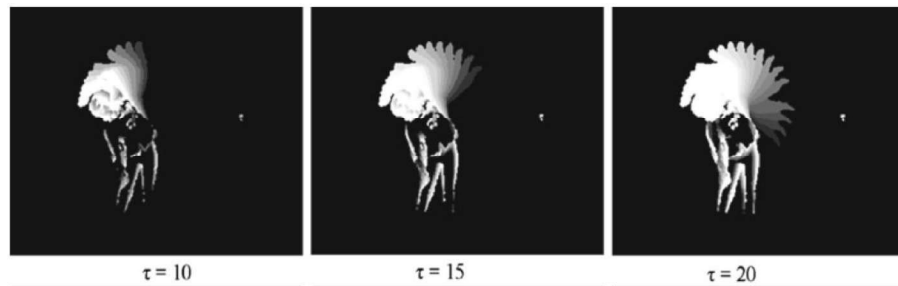


Figure 2.15: Effect of different values of τ [71]

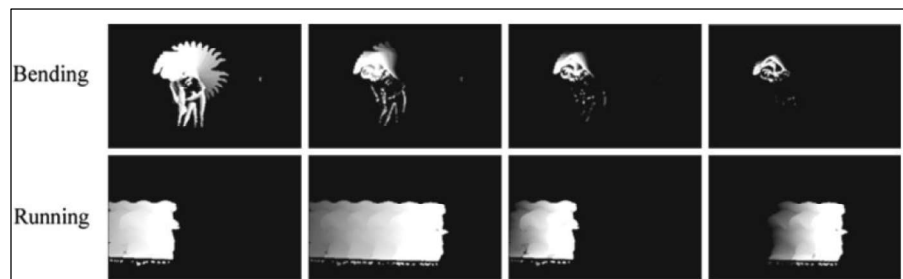


Figure 2.16: Actions with different δ values. Top row – bending action with different δ values, from left to right, the values are 1, 3, 5, and 10 respectively. Bottom row – running action where the first left 2 images are $\delta = 1$, whereas the latter two are $\delta = 3$ [71].

The update function can be generated by methods based on background subtraction, frame-to-frame differencing methods, and optical flow. Some works that generate MHI from background subtraction, such as [72] and [73]. Some issues with background subtraction are the need for the background to be static, and factors such as the outdoors and clutter should be absent.

Some works use frame-to-frame differencing methods, such as [74]. However, poor extraction of motion pixels can be seen in Figure 2.17 [71]. One caveat for the frame-to-frame differencing method is the importance of choosing a threshold for binarization.



Figure 2.17: Effects of unsuitable threshold value. An inappropriate threshold value will cause noise in the frame difference, which would cause noise in the final MHI. The threshold value is set at 30, 50, 74, and 150 from left to right, respectively. A noisy background is noted with a threshold value of 30, yet some motion information is missing with a higher value [71].

Some works that use optical flow for the generation of MHI include [75], [74], and [76]. Particularly, Ahad et al. [75] use optical flow's four channels to generate MHI. A gradient-based optical flow vector is calculated between 2 frames and split into four channels, as can be seen in Figure 2.18 [75]. Although optical flow is more robust with camera motion and variable background, it is computationally expensive and sensitive to noise and texture. Another problem is that optical flow is not robust with very low-resolution frame sequences.

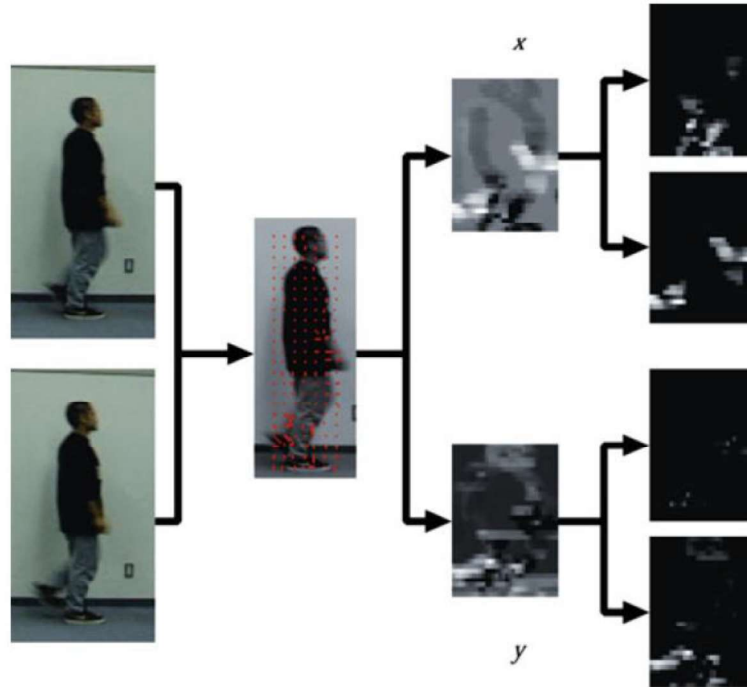


Figure 2.18: Four different channels of optical flow are used to generate directional MHI [75].

One of the key limitations of the MHI method is its inability to perform well in the presence of motion overwriting due to self-occlusion, as can be seen in Figure 2.19 [71].

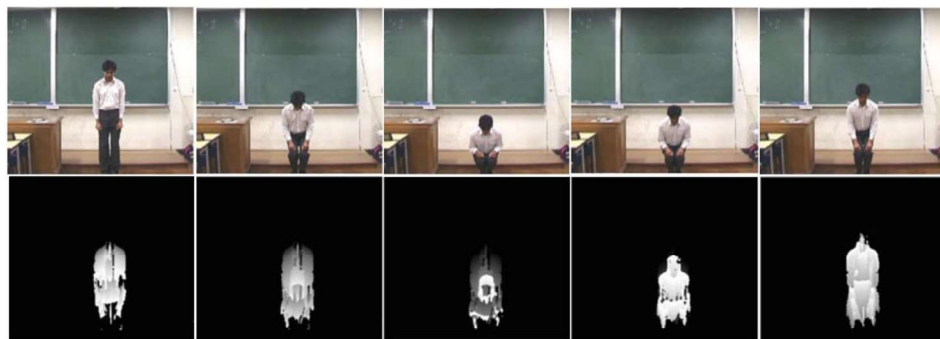


Figure 2.19: Motion overwrite issue for sitting down and standing up motion (self-occlusion) [71].

Valstar et al. [77] proposed a method that would help overcome the motion overwrite issue. They used a multiple-level MHI (MMHI), which records motion history at several time intervals. The number of history levels is represented as n . Therefore, a sequence of images has $(n + 1)$ frames.

Then, to encode the motion that occurs at different times in instances that are at the same location, so that it can be uniquely decoded later, a simple bit-wise coding scheme is used. If motion happens at time t at position (x,y) , $2^{(t-1)}$ is added to the old value of MMHI. The equation of MMHI can be represented by the equation below:

$$MMHI(x, y, t) = MMHI(x, y, t - 1) + \Psi(x, y, t) * 2^{t-1}$$

with $MMHI(x, y, t) = 0$ for $t = 0$ (2-2)

Although MMHI produces lower recognition rates than conventional MHIs, the authors believe that MMHI offers benefits in conditions where motion self-occlusion is common.

Another proposed method that helps with the motion overwriting problem is the one proposed by [75] that was explained before. Optical flow is split into four channels to get four-directional motion templates.

Meng et al. [78] solve the overwriting problem using a method called the hierarchical motion history histogram (HMHH). If motion occurs at frame k at pixel (u, v) , then $D(u, v, k) = 1$, else $D(u, v, k) = 0$. For pixel (u, v) , the

motion mask $D(u, v, :)$ of the particular pixel is a binary sequence, which can be expressed as the equation below, where $N + 1$ is the total number of frames:

$$D(u, v, :) = (b_1, b_2, \dots, b_N), b_i \in \{0, 1\} \quad (2-3)$$

Patterns P_i is defined in the motion mask ($D(x, y, :)$) sequences, based on the number of connected components ‘1’. That is,

$$P_1 = 010, P_2 = 0110, P_3 = 01110, \dots, P_M = 01 \dots 10 \quad (2-4)$$

Then, a sub-sequence is defined as $C_i = b_{n1}, b_{n2}, \dots, b_{ni}$, and the set of all sub-sequences of $D(u, v, :)$ is denoted as $\Omega\{D(u, v, :)\}$. Then, the number of occurrences of each specific pattern P_i in the sequence $D(u, v, :)$ is counted for each pixel (u, v) , as shown in equations 2-5. Refer to Figure 2.21 and Figure 2.22, and in the equation below, $\mathbf{1}$ refers to the indicator function.

$$HMHH(u, v, P_i) = \sum_j \mathbf{1}\{C_j = P_i | C_j \in \Omega\{D(u, v, :)\}\} \quad (2-5)$$

A greyscale image can be built from each pattern P_i . This is the Motion History Histogram (MHH). With all the patterns $P_i, i = 1 \dots M$ together, collectively, they make the ‘‘Hierarchical Motion History Histogram’’ (HMHH) representation. Figure 2.20 presents the whole algorithm.

Algorithm (HMHH)

Input: Video clip $f(u,v,k)$, $u=1,\dots,U$, $v=1,\dots,V$, frame $k=0,1,\dots,N$
Initialisation: Pattern M , $\text{HMHH}(1:U,1:V,1:M)=0$, $I(1:U,1:V)=1$
For $k=1$ to N (**For** 1)
 Compute: $D(:, :, k)$
 For $u=1$ to U (**For** 2)
 For $v=1$ to V (**For** 3)
 If Subsequence $C_j=\{D(u,v,1),\dots,D(u,v,k)\}=P_i$
 Update: $\text{HMHH}(u,v,P_i)=\text{HMHH}(u,v,P_i)+1$
 End If
 Update: $I(u,v)$
 End (For 3)
 End (For 2)
End (For 1)
Output: $\text{HMHH}(1:U,1:V,1:M)$

Figure 2.20: HMHH Algorithm, extracted from [78].

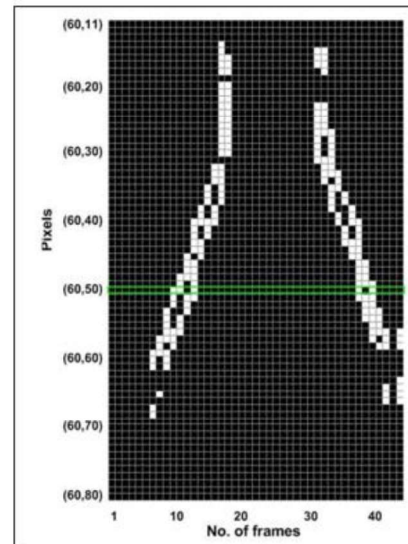


Figure 2.21: Left image: MHI of a handwaving action. Right image: is $D(:, :, :)$ of the red line of the left image. Each row is $D(u, v, :)$ represented as a fixed pixel (u, v) . The green line is the motion mask of pixel $(60, 50)$, through time, denoted as $D(60, 50, :)$ [78].

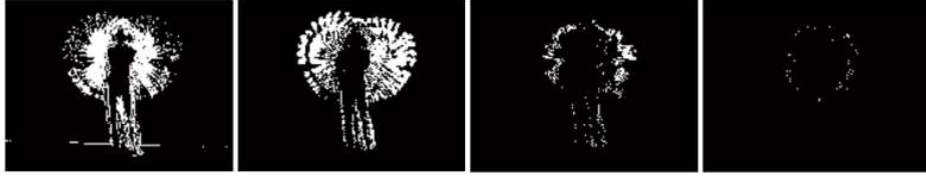


Figure 2.22: Samples of an HMHH. From left to right, there are four patterns - $HMHH(:, :, P_1)$, $HMHH(:, :, P_2)$, $HMHH(:, :, P_3)$, $HMHH(:, :, P_4)$ [78].

However, despite the HMHH solving the motion overwrite problem, in terms of usability, it is not more robust than the usual MHI methods.

2.4.2 MHI AND DNN

Recent works integrating MHI with CNN models are done by Chandragiri and Ijjina [79], where they combined the spatial and temporal information across video frames by training a DNN (VGG16 or ResNet152) classifier to recognize actions from MHI representation. Techniques such as transfer learning of pre-trained models are also used. The proposed flowchart in [79] is simple and can be seen in Figure 2.23.

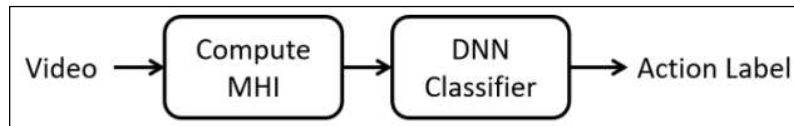


Figure 2.23: MHI is computed using an OpenCV library, and DNN is used to produce either one of the 4 action labels (None, Taking Money, Picking Up Receipt, Giving Receipt) [79].

Sincan and Keles's [80] work is for sign language recognition. The authors proposed an RGB-MHI image, which represents a summary of each video in a single frame. Then, a model (RGB-MHI model) is proposed that learns from these single frames a representation of relevant spatial and motion

patterns. Then, two different approaches using the RGB-MHI model are proposed. The first: RGB-MHI model is used as a motion-based spatial attention module that is integrated into a 3D-CNN architecture. The second: Late fusion technique with RGB-MHI and 3D-CNN features.

The RGB-MHI image is made by splitting a video stream into 3 equal portions, motion histories are then calculated for each portion independently. To construct motion histories, a motion history image (MHI) needs to be created first. MHI is generated by summing the absolute values of consecutive frame differences – (2-6)

$$M(i, j) = \sum_{t=2}^N |I_{t-1}(i, j) - I_t(i, j)|W \quad (2-6)$$

N denotes the number of frames in a video, I_t denotes the t^{th} video frame, (i, j) represents the image pixel coordinates, and W denotes the weight for the absolute difference, where $W = t/N$.

After the motion histories are generated, a 3-channel colored MHI image (called star RGB [80]), is created – Figure 2.24. In this representation, the B-channel contains the motion history information from the first temporal region, the G-channel has the motion history information from the central one, and the R-channel is from the last part.

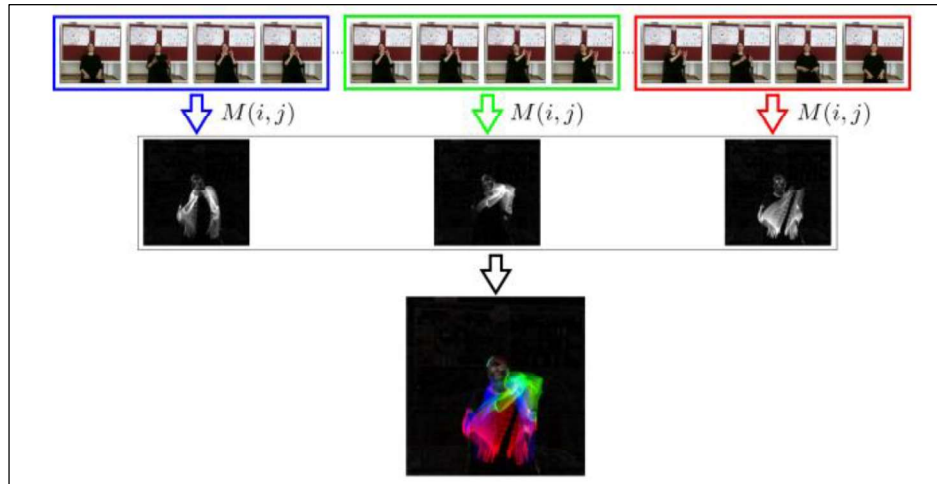


Figure 2.24: Flowchart of the creation of an RGB-MHI [80].

The RGB-MHI model can be used in 2 proposed ways, 1) RGB-MHI model is used as a motion-based spatial attention module that is integrated into a 3D-CNN architecture; and 2) Late fusion technique with RGB-MHI and 3D-CNN features. A flowchart summary is shown in Figure 2.25, and a summary of their results is shown in Table 2.4. The metric they used to measure performance is accuracy – (2-7), the results show that RGB-MHI helps with an improvement in accuracy by ~ 1% compared to when it is not used.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2-7)$$

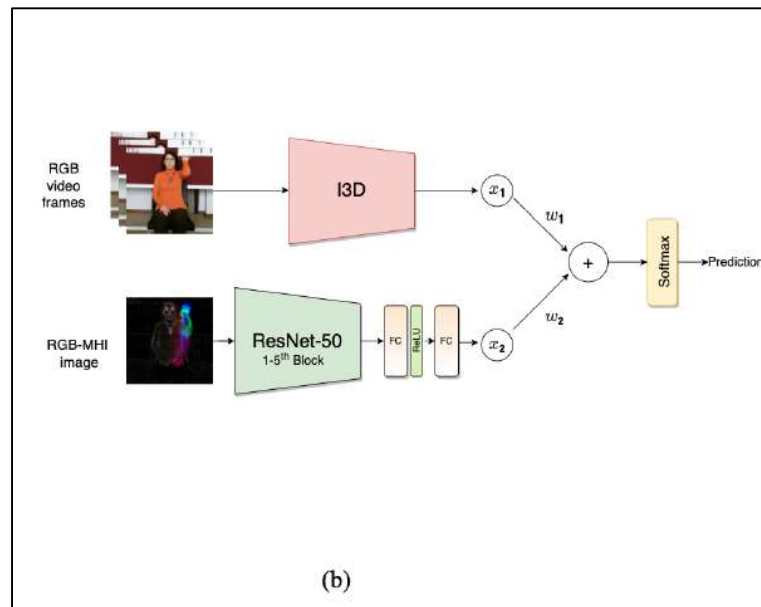
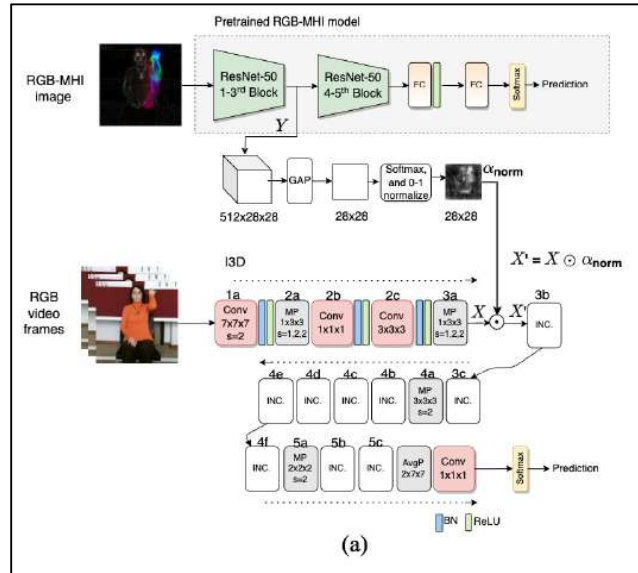


Figure 2.25: Proposed methods [80]. a) I3D with RGB-MHI attention; b) I3D + RGB-MHI fusion

Table 2.4: Results of the proposed method [80]

Method	Test Accuracy (%)
I3D	89.30
I3D with self attention	89.94
I3D with RGB-MHI attention	90.18
I3D + RGB-MHI fusion	91.13
I3D with RGB-MHI attention + fusion	91.55

One caveat of the proposed method [80] is that if motion overwrite exists within any three parts of the video sequence, information will still be lost. Moreover, if the motion occurrence's time is not known, it is difficult to set the length and number of frames, or how many parts a video should be split into.

Toudjeu and Tapamo [81] proposed a method that can be summarized into the flowchart in Figure 2.26. Their innovation lies in the first step, the preprocessing step, where the MHI is generated. After that, when the MHI is formed, the recognition stage, which involves a 2-D CNN and a Fully Connected Network, is used to classify the action. To elaborate on the first step, the video sequence is processed to a single image, known as the MHI template. This template keeps a history of the temporal changes at each pixel location, which decay over time. The equation can be seen in (2-8) – (2-10), where (2-9) is the update function and (6-3) is the difference of frames. Parameters τ, λ, δ and ϵ denote the duration of frames, decay parameter, interframe distance, and difference threshold, respectively.

$$M_{\tau}(x, y, t) = \begin{cases} \tau & \text{if } U(x, y, t) = 1 \\ \max(0, M_{\tau}(x, y, t) - \lambda) & \text{otherwise} \end{cases} \quad (2-8)$$

$$U(x, y, t) = \begin{cases} 1 & \text{if } \Delta(x, y, t) \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (2-9)$$

$$\Delta(x, y, t) = |I(x, y, t) - I(x, y, t \mp \delta)| \quad (2-10)$$

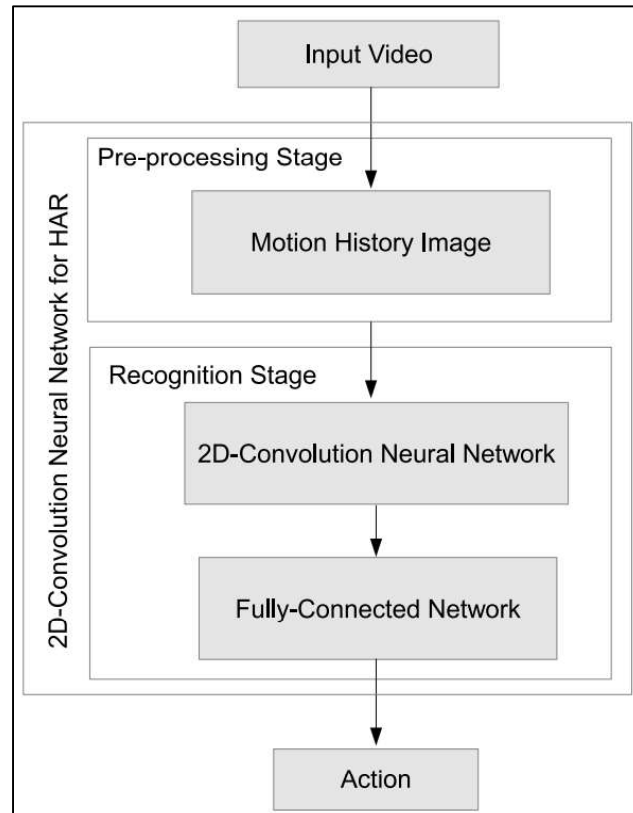


Figure 2.26: Proposed methodology [81].

There are a few variants of MHI that try to solve the overwriting problem that is detailed above. However, the solutions have issues. In summary, [75] and [76] are both computationally expensive; Valstar et al. [76] and Meng et al. [78] solve the overwriting problem. The overwriting problem is usually present in general MHI, where movement information is lost due to overlapping in the same pixel location. However, it is reported that [77] [78] does not improve recognition results more than the general MHI; additional design, in addition to the proposed method in [77] [78], is needed to improve recognition results. Objectively, compared to the algorithms, the proposed method should be computationally inexpensive and be able to fully encode without information loss the whole history of a designated time span.

In order to cope with the complexity of an unconstrained underwater environment and to make use of temporal information from the underwater videos, similar to the previous works, we adopt an extension of the motion history images (MHI) concept to complement the YOLOv5 architecture. This project proposes 2 types of motion history representation, which explore blending and overlapping hue values to create motion history patterns. Color information helps distinguish motion overlap. Then, the first type of motion history representation uses addition on the frame differences, while the second type uses the overwriting concept. In order to prove the effectiveness of the proposed trail formulation method, further experiments are done by replacing the proposed method with the traditional MHI method in Chapter 4. Currently, to the best of our knowledge, no MHI method is used with fish detection or the detection of multiple separate objects. Therefore, this project explores the possibility of using MHI in conjunction with DNNs for this fish detection task.

CHAPTER 3

PROPOSED METHOD

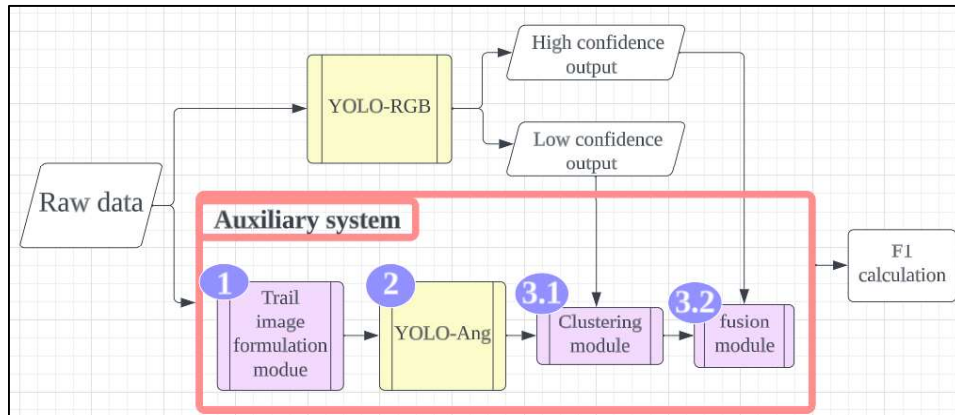


Figure 3.1: System flowchart of our proposed method

The proposed method in Figure 3.1 introduces an auxiliary system that generates additional bounding boxes and increases the recall rate. This auxiliary system provides information to select low-confidence bounding boxes produced by YOLOv5, and thus it produces additional candidates (bounding boxes) for reducing FN probability. The auxiliary system is trained on a dataset produced by the Trail Image Formulation module (labelled as 1). The detector of the auxiliary system is a modification of YOLOv5, and we name it YOLO-Ang (label 2); the “Ang” represents Angle, as YOLO-Ang is trained to predict angles associated with the output bounding box. The output from YOLO-Ang is then merged using a clustering module (label 3.1) and a simple fusion module (label 3.2). The following sections detail each part of the system and the motivation behind the design.

3.1 TRAIL IMAGE FORMULATION MODULE

From observation, fishes usually swim in a simple and straight trajectory; however, there are occasions when fishes sometimes swim in erratic movements. Based on this observation, using frame differences between only two frames is sometimes not sufficient to determine the movement of a fish. Some of the circumstances are: (a) when fish take a short break and are kept still in between frames; this is shown in Figure 3.2; (b) when they flip around, thus making it visually rare for YOLOv5 to learn its features; this is demonstrated in Figure 3.3; or (c) when they swim to a background that is similar to their colour, as shown in Figure 3.4. Instance (a) can usually be detected by YOLOv5 except for some rare occasions; however, it would be more difficult for instances (b) and (c).

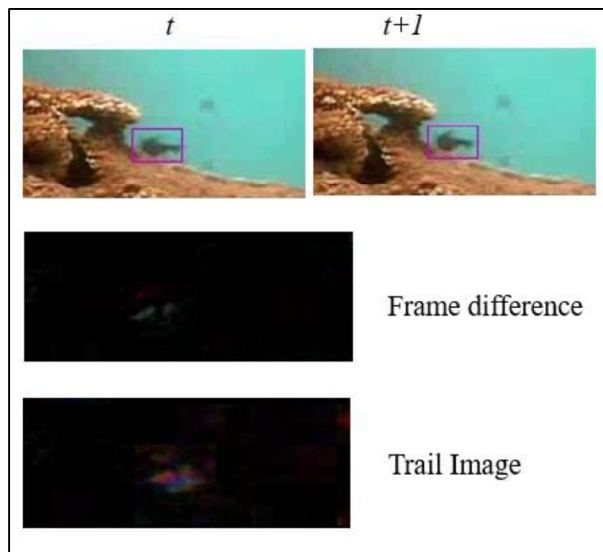


Figure 3.2: Trail image subjectively highlights the movement of fairly still fishes better than frame difference.

For Figure 3.2, we can see the target fish does not move much from frame t to $t+1$. Subjectively, trail images are able to highlight the movement of the target fish better since more frames (information) are used to generate the result. The construction of the trail image is to be elaborated on in Section 3.1.1.

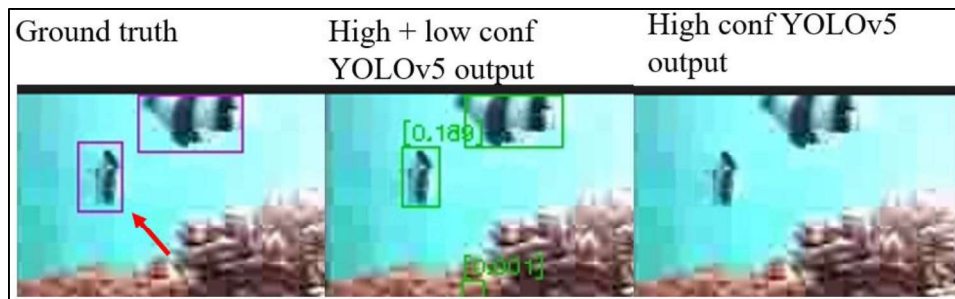


Figure 3.3: YOLOv5 gives low confidence to flipping fish.

In Figure 3.3, the left picture shows the ground truth, and the red arrow is pointing at the target fish instance that demonstrates the flipping fish issue. The middle frame shows both low and high confidence YOLOv5 outputs, and the right column shows only the high confidence ones. As can be seen here, the fish is at an angle to the camera, which is rare in the training set, and thus YOLOv5 does not consider this prediction highly confident.

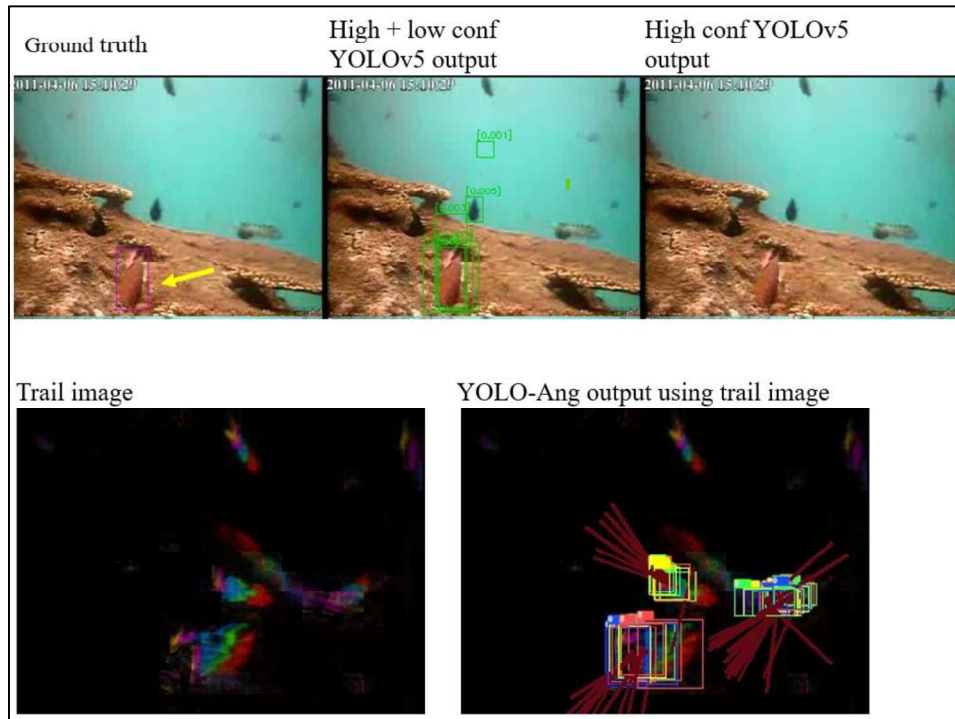


Figure 3.4: Top: YOLOv5 inability to assign a high confidence score to camouflaged fish. Bottom: Boxes from YOLO-Ang output act as a complementary system to support low-conf YOLOv5 output.

Therefore, one concept that helps in solving this problem is collecting information from multiple frames. One example is Motion History Image (MHI). According to [16], MHI is useful in low illumination conditions; such a condition can also be applied to objects that have a similar appearance to their background. The MHI representation encodes the time information (moving parts and motion flow) of multiple frames into a single frame. The proposed Trail Image Formulation module is based on the concept of MHI, where time information from multiple frames is encoded into a single frame; however, there are some major differences. One of the differences is that, MHI uses binarization to determine the pixels with motion vs. pixels without motion; information about the pixel values of the frame differences is ignored. Trail images keep the pixel values of the frame difference. This property of trail images helps preserve

information that would otherwise be lost using MHI. Retaining the pixel values of the frame difference can retain some information about fish textures.

Here, we propose 2 types of trail image formulation: subchapter 3.1.1.1 uses addition to combine multiple numbers of frame differences, while subchapter 3.1.1.2 uses overlapping concepts to combine multiple numbers of frame differences.

3.1.1 TRAIL IMAGE CONSTRUCTION

3.1.1.1 CONSTRUCTED WITH ADDITION

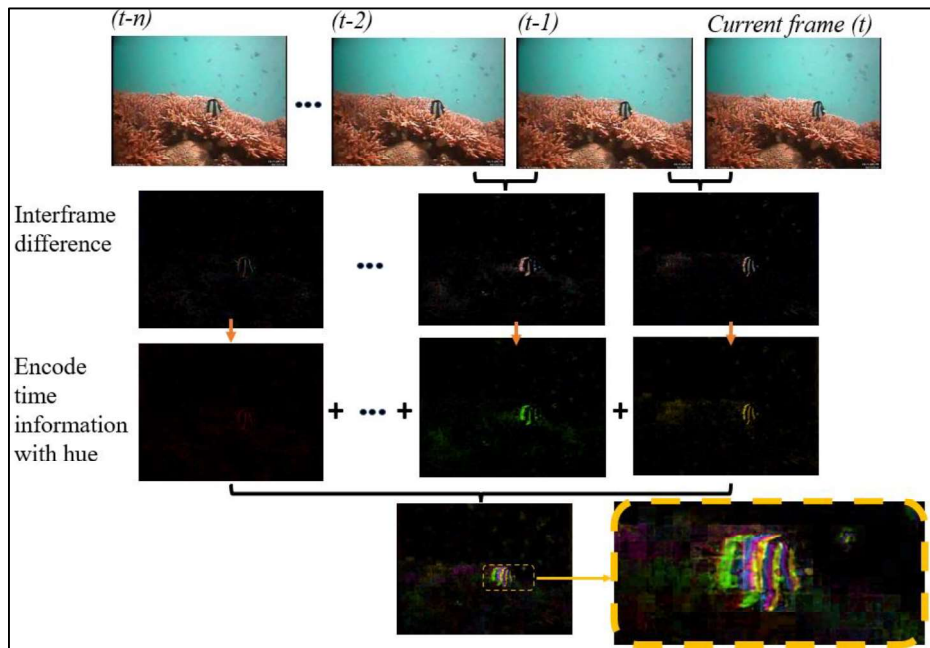


Figure 3.5: Trail image construction flowchart

Figure 3.5 shows how the trail image of the current frame t is created. Firstly, the frame differences are calculated by finding the difference between two consecutive frames. Clipping to 0 is made for any pixel that is lower than 0. For example, 4 original frames produce 3 frame differences. The frame difference (FD), initially in RGB format, is then converted to the HSV colour space. The conversion formula to convert RGB to HSV is in Equation (3-1), while the conversion formula from HSV to RGB is in Equation (3-2) [74]. Note that in the subsequent equations, the S and V values are normalized to 255, that is, the range of both S and V is between 0 and 255.

$$\begin{aligned}
 V &= \max(R, G, B) \\
 S &= \begin{cases} \frac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
 H &= \begin{cases} \frac{43(G - B)}{V - \min(R, G, B)} & \text{if } V = R \\ \frac{85 + 43(B - R)}{V - \min(R, G, B)} & \text{if } V = G \\ \frac{170 + 43(R - G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases} \quad (3-1)
 \end{aligned}$$

If $H < 0$, then $H \leftarrow H + 255$. On output $0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 255$.

Given $0 \leq H \leq 255, 0 \leq S \leq 1$ and $0 \leq V \leq 1$:

$$\begin{aligned}
 X &= C \times \left(1 - \left\lfloor \left(\frac{H}{43} \right) \bmod 2 - 1 \right\rfloor \right) \\
 m &= V - C
 \end{aligned}$$

$$(R', G', B') = \begin{cases} (C, X, 0), 0 \leq H < 43 \\ (X, C, 0), 43 \leq H < 85 \\ (0, C, X), 85 \leq H < 128 \\ (0, X, C), 128 \leq H < 170 \\ (X, 0, C), 170 \leq H < 213 \\ (C, 0, X), 213 \leq H < 255 \end{cases} \quad (3-2)$$

$$(R, G, B) = ((R' + m) \times 255, (G' + m) \times 255, (B' + m) \times 255)$$

Then, for each FD frame, the S (saturation) is set to 255. According to the number of FDs (indicated by n) encoded in the final trail representation, the H (hue) part of the frame is set to $(255/n) * d$, where d represents the local index of FD frames counted backwards from the current FD frame with time index t . For example, if we want to detect the number of fish at FD frame t and we use 3 FD frames, $f_D(t-2)$, $f_D(t-1)$, $f_D(t)$, to generate a trail image. In this case, the local index value $d \in \{0,1,2\}$. The current FD index is $d=0$. The frame that is two frames before frame t has a time index as $(t-d)$ with $d=2$. Note that the FD frame time index is derived from the original image frame index because $f_D(t) = \max(0, (f(t-1) - f(t)))$, where $f(t)$ is the original frame with time index t .

The V (value) of the FD frames is preserved. Then, these processed FDs are converted back to the RGB colour space and added together to produce a single representation frame, with clipping from 0 to 255 values, the range of an 8-bit pixel. Figure 3.6 show the processing steps. The trail image now encodes the object trajectories of several frames into a single image.

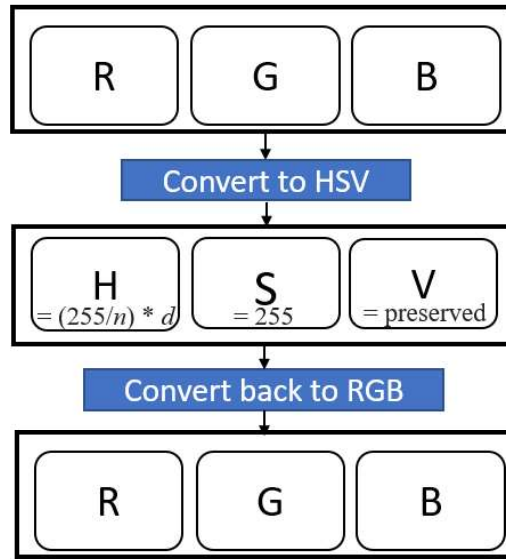


Figure 3.6: Steps to encode time information for each frame difference, before added together

3.1.1.2 CONSTRUCTED WITH OVERLAP

For trail formulation using overlap concepts, the first step is similar to trail formulation with addition, where frame differences (FDs) are calculated by finding the difference between two consecutive frames, and the pixel values are clipped to 0. Then, the RGB frame differences are converted into the HSV domain using Equation (3-1).

From here, the H and V channels go through a different type of processing than the trail formulation with addition. The S channel stays the same with a value of 255. For the H channel, the conventional MHI formula is used, Equation (3-4). Where $D(x, y, d)$ represents the frame difference (FD) that has been converted to grayscale, n represents the number of frame differences used to form a single trail image, and d is the local index defined earlier.

$$D'(x, y, d) = OTSU_{thresh}(D(x, y, d)), \text{ where } d = \{n - 1, \dots, 1, 0\}$$

$OTSU_{thresh}(\cdot)$ is the binary Otsu's thresholding method

(Kurita, Otsu and Abdelmalek, 1992)

$$H(x, y, d) = \begin{cases} 0, & \text{if } D'(x, y, d) = 0 \\ \frac{255}{n} * d, & \text{otherwise} \end{cases}$$

$$H_n(x, y) = \max (\{H(x, y, d), d \in \{n - 1, \dots, 1, 0\}\}) \quad (3 - 4)$$

The V channel also use the overlapping concept; however, instead of assigning a time-coded magnitude, the original value of the V channel is preserved, Equation (3-5). Note that the recursion of Equation (3-5) starts from $d=n$, and then $d=n-1, \dots$

$$V'(x, y, d) = \begin{cases} V(x, y, d) & \text{if } D'(x, y, d) = 1 \\ V'(x, y, d + 1) & \text{otherwise} \end{cases}, d = n - 1, \dots, 1, 0$$

where initial values: $V'(x, y, d = n) = 0$

$$V_n(x, y) = V'(x, y, d = 0) \quad (3 - 5)$$

In the end, the H_n , $S (=255)$, and V_n channels are merged and converted back to the RGB domain with clipping. Chapter 4 will detail the performance difference between trail formulations with addition and overlap.

3.1.2 GROUND TRUTH FOR TRAIL IMAGE

In addition to the “trail” representation, the angle information is also calculated to improve detection. Inference output with angle is to be used in the post-processing phase, which consists of the clustering module and the fusion module.

Similar to [13], [14], and [15], where MHIs are used in conjunction with CNNs, the trail image is to be used by YOLO-Ang. However, MHI is typically used to recognize “action”, and often there is only one object per frame. Compared to our proposed method, trail images have multiple objects per frame, and the purpose of constructing trails is to improve object detection rates. In other words, we first detect the trail of an object, and then we are able to track the object. This is the “track to detect” concept discussed in several detect-and-track papers.

For training YOLO-Ang, the ground-truth of the angle needs to be created. The ground-truth set includes not only the bounding box coordinates but also the angle of fish direction for every box. The angle is derived from the ground-truth bounding boxes of two successive original frames. Different classes are assigned to the bounding box according to the index of frames counted from the current frame. So, if 4 original RGB frames are used, the ground truth boxes and angles of all 4 original RGB frames are included in the ground truth of the trail image. The current frame’s boxes are class 0, and the boxes of the previous 3 frames are classes 1, 2, and 3. Therefore, for example,

the previous frame is class 1. Radians are used as the unit of angle.

Ground-truth angles are created by measuring the angle of the vector, which is defined by the center of the bounding box of a fish in the current frame and the center of the bounding box of the same fish instance in the previous frame. We find the previous frame's same fish instance by calculating the distance between all the fish instances from the previous frame and the current frame fish instance, and then choose the one with the shortest distance. Referring to Figure 3.7, the arrow (with an angle) points to the previous frame fish instance, which is the same fish instance as the current frame. Manual checking shows that it is able to correctly find the same fish instance from the previous frames; no cases of incorrect matching have been found so far. If there is no ground truth data available for the previous frame, there will be no angle for the bounding boxes of that frame. The trail image and its ground truth with angle will then be used to train and test the modified YOLOv5 called YOLO-Ang in Section 3.2.

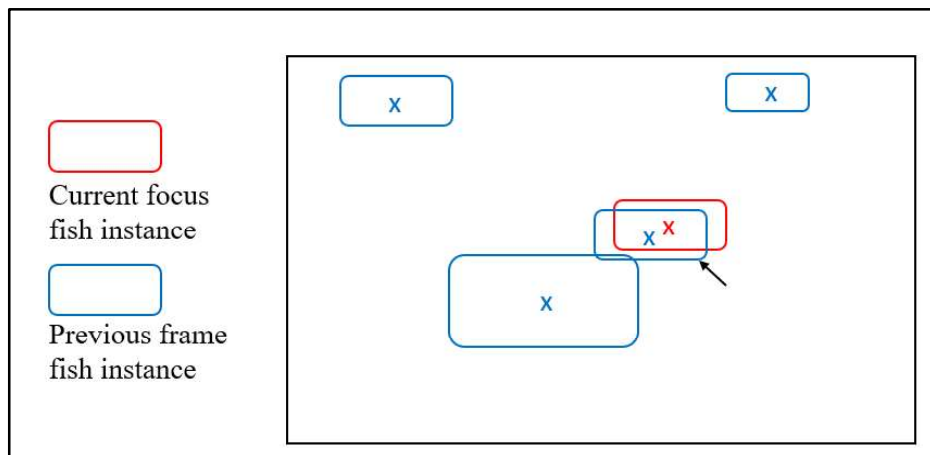


Figure 3.7: Illustration of determining the same fish instance from the previous frame

3.2 YOLOV5 WITH ANGLE

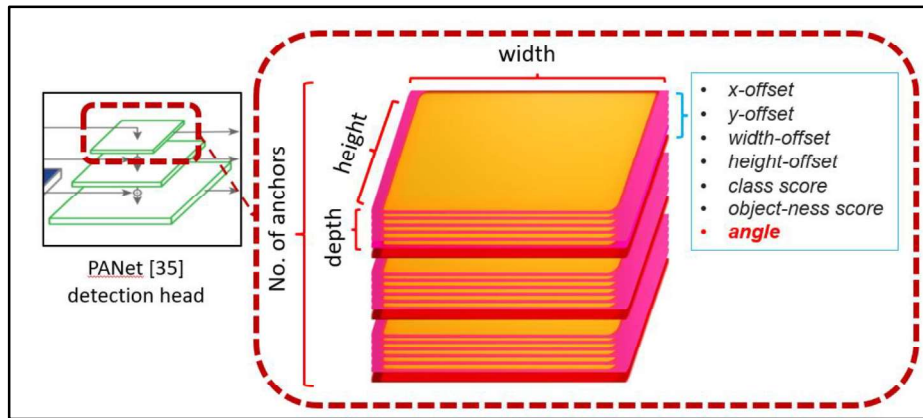


Figure 3.8: Illustration of a single layer of the detection head

The original YOLOv5 produces bounding boxes and its confidence scores as output. We modify YOLOv5 to generate the angle information and call it “YOLO-Ang” (YOLOv5 with Angle).

As mentioned in Section 2.2, YOLOv5 uses PANet [31] as its detection neck. An illustration of PANet and one of its feature maps are shown in Figure 3.8. Its 3-level feature map is called the *detection head* and is responsible for 3 different spatial resolutions; these feature maps will each undergo further convolution, which is part of the detection process. Detection will then be performed directly on these three levels of detection heads. For each level of the detection head, there are 3 dimensions: width, height, and depth. The width and height represent spatial resolution, and each pixel has a certain receptive field in the original image. Each pixel has its own depth. The original YOLOv5 depth is $(a * (5 + n_{cls}))$ layers deep, where a represents the number of anchors per pixel, and the 5 values represent: *x-offset*, *y-offset*, *width-offset*, *height-offset*, and

object-ness score. N_{cls} is the number of classes, and each class will have its own score. For the proposed method, YOLO-Ang, we added an additional layer for the depth; this layer will be in charge of learning angle data. Therefore, YOLO-Ang depth is $(a * (6 + n_{cls}))$ layers deep. This is illustrated in Figure 3.8: yellow layers represent layers that existed in YOLOv5, while the highlighted red layers are added for the proposed method. In total, there are 3 extra layers that are in charge of learning angle information if the number of anchors is set to 3. The red letters in Figure 3.9 below show the changes the proposed method made.

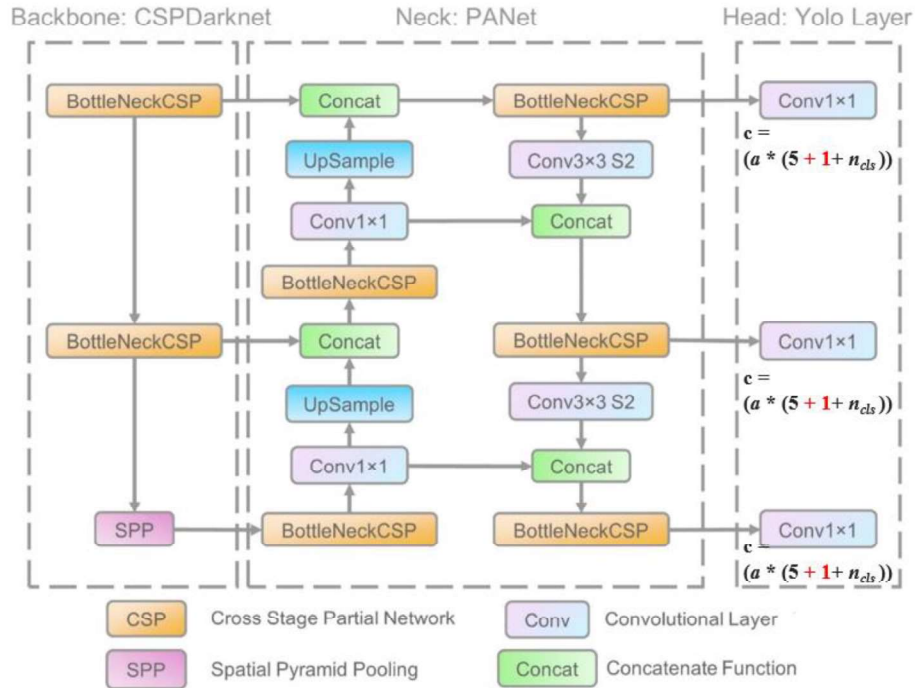


Figure 3.9: Architecture of YOLOv5, figure extracted from a paper by Dima and Ahmed [83]

YOLOv5 originally uses 3 types of loss: box loss, object-ness loss, and class loss. Bounding box loss is in the form of an IOU between the prediction box and the target box. The IOU can be defined as $IoU(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$,

where B_p represents bounding box prediction, and B_{gt} represents ground truth bounding box. Object-ness loss uses the binary cross entropy with logits loss using the ground truth bounding box as its target. Class loss also uses binary cross entropy with logits loss using ground truth class as the target. Binary cross entropy loss can be defined as: $loss = -p \cdot \log(\hat{p}) - (1 - p) \log(1 - \hat{p})$, where p is the true probability and \hat{p} is the predicted probability.

YOLO-Ang has an additional loss, which is the angle loss. It uses the mean square error loss: $(\hat{\theta} - \theta)^2$ to calculate the difference between the target angle, θ , and the predicted angle, $\hat{\theta}$ (in normalized radians). The ground truth bounding box that does not have an angle will be omitted from the angle loss calculations. Mean square error is used due to the fact that it is the simplest loss to measure a target that has a continuous value (non-discrete value).

The weighting parameters for the losses are as follows: box loss = 0.05, object-ness loss = 1.0, class loss = 0.5, and angle loss = 0.5. These parameters have been set as default by the authors of YOLOv5, and these parameters are generally used. The reason why box loss has such a low contribution is because it considers every single box's anchor box. Object-ness loss represents box loss for matching boxes with targets; therefore, it needs more weight. Therefore, it is reasonable to assign angle loss weigh similarly to class loss, which is half of object-ness loss.

The inference outputs of YOLO-Ang are bounding box coordinates that mark the existence of trails in an overlapping manner. For each bounding box,

there is an angle value assigned to it, which represents the trajectory of the trails. These outputs will then be processed by the modules below.

3.3 POST-PROCESSING MODULES

The outputs from YOLO-Ang act as a support for filtering the low-confidence YOLOv5 outputs; however, further processing is needed to make them useful. The post-processing modules take the outputs from YOLO-Ang and merge them with the original YOLOv5 output. These modules are handcrafted and have some manual parameters that can be adjusted.

The post-processing modules are designed based on the concept of “majority voting” since there are a large number of bounding boxes and their corresponding angles. Therefore, the error rate (compared to the ground truth) per inference box can be high. The post-processing modules can be separated into 2 modules, the clustering module, and the fusion module. Below are the details of these mechanisms.

3.3.1 CLUSTERING MODULE

As shown in Figure 3.10 below, the clustering module generates a few “support boxes”, which will then later be used for the next module, which is the fusion module. The fusion module matches the support boxes with the low-confidence boxes produced by the original YOLOv5.

The YOLO-Ang outputs (bounding boxes) indicate the existence of “trails” in the trail image, as shown on the left of Figure 3.11. However, the output bounding boxes and angle data contain errors because they cannot distinguish well between noise and fish. Therefore, some extra post-processing steps are needed to produce the desirable support boxes. We develop a clustering-like algorithm to extract meaningful information.

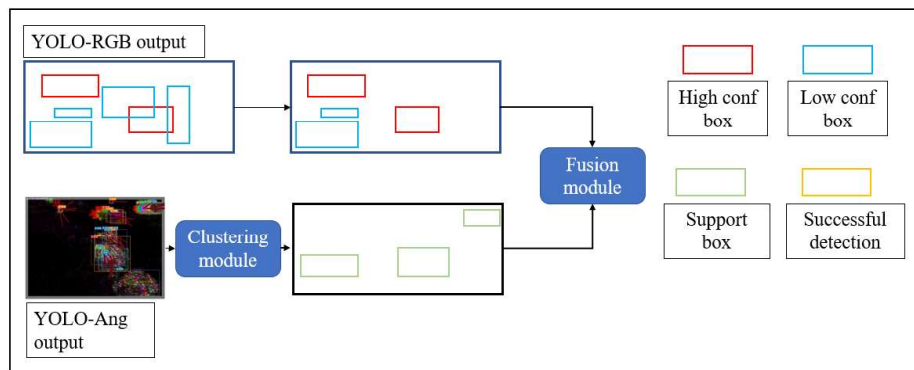


Figure 3.10: The clustering module is responsible for processing the YOLO-Ang output to generate support bounding boxes, which are then used by the fusion module

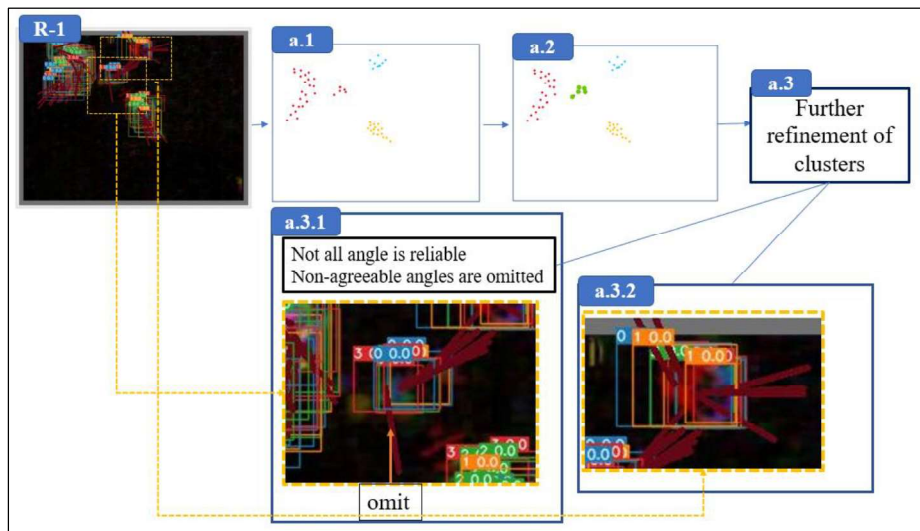


Figure 3.11. Samples of the YOLO-Ang outputs. Multiple fish objects are shown in this image. Four steps are designed to generate the final support boxes.

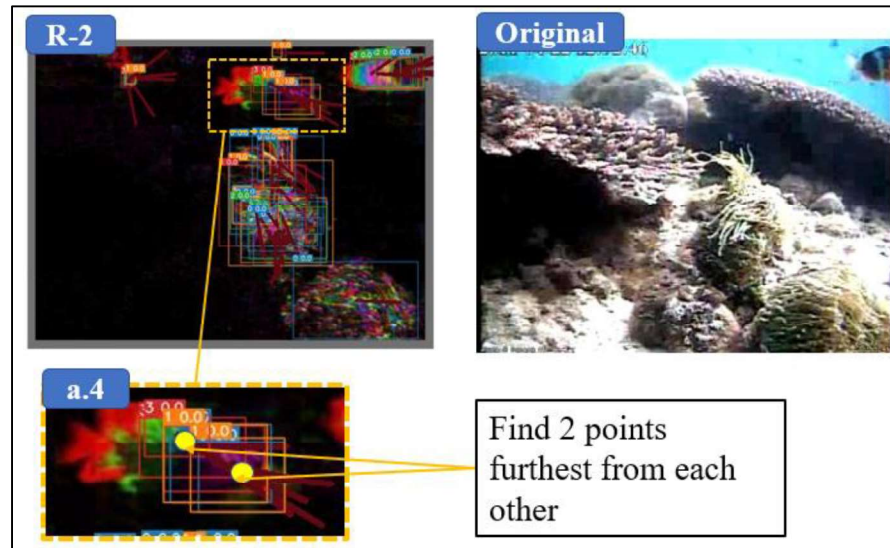


Figure 3.12: Another sample of YOLO-Ang output. An instance of a fish object is shown to illustrate the operation of Step a.4. The original RGB frame is also shown here for reference.

Figures 3.11 and 3.12 show the steps in the clustering module; not all details are included to simplify the figures. Images R-1 and R-2 show two YOLO-Ang output trail images, which highlight different issues that the clustering module is able to overcome. The purpose of clustering module is to produce final clusters (of bounding boxes) that correspond to the trails or traces of fish. We will describe the detailed steps of the clustering module in the following paragraphs.

Set B is the YOLO-Ang output after removing boxes that have a width or height lower or higher than the set threshold parameters a_1 , a_2 respectively. Set C is the set of clusters, and set C_j is a single cluster made up of elements from B . B_{seed} represents a randomly selected element in set B , which will be used to initialize a cluster.

A more detailed explanation of each sub-module of the clustering module and its 8 parameters (a_1 , a_2 , a_3 , a_4 , a_5 , a_6 , and $IOUcutoff$, $Confcutoff$) will be explained here. The output of YOLO-Ang consists of bounding box coordinates for multiple classes with an angle associated with each box. In **Algorithm a.1** below, boxes are first omitted if their length or width are above and below the selected lower (parameter a_1) and lower limits (parameter a_2). (The default values are 0.05 and 0.5 of the frame size). This is to first filter out boxes that are unlikely to contain any useful information. Then, the distance between the centers of the bounding boxes is calculated. Boxes within a set distance range (parameter a_3 , default value 40 pixels) are assigned to the same cluster as shown in **Algorithm a.1** below. A simple set distance range is used instead of the more popular K-means clustering method because we do not know the value of clusters we would end up with. Having a set distance range allows flexibility in determining the number of clusters. Clusters that are less than 3 boxes are ignored in the next processes since they are most likely noise. This decision was made after observing a few frames during the construction of this algorithm.

Algorithm a.1 (clusters construction)

The i^{th} bounding box is represented by $(xywh_i)$,
which means its center coordinates are (x_i, y_i)
and its width and height are w_i and h_i , respectively.

Euclidean distance between 2 boxes:

$$distance_1(i, j) \equiv \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$|A| \equiv$ number of elements in set A

Input: $B = \{(xywh_i) | a_{-1} < w_i < a_{-2} \text{ and } a_{-1} < h_i < a_{-2}\}$

$j = 1$

Step 1

$B_{seed} =$ the set containing a randomly selected box inside B

$C_j = \{(xywh_k) \in B | (xywh_s) \in B_{seed}, distance_1(s, k) < a_{-3}\}$

Step 2

$C_t = \{(xywh_k) \in B | \exists (xywh_i) \in C_j, distance_1(i, k) < a_{-3}\}$

If $(|C_t| \neq 0)$, $C_j \leftarrow C_j \cup C_t$

Step 3

$B \leftarrow (B - C_j) \equiv \{(xywh_i) \in B \text{ and } (xywh_i) \notin C_j\}$

If $(|B| = 0)$, **Stop**; goto **Output**

Else if $(|C_t| > 0)$, goto **Step 2**

Else if $(|C_j| < 3)$, C_j is not a valid cluster. Remove C_j , goto **Step 1**

Else C_j is completed. $j \leftarrow j + 1$, go to **Step 1**

Output: $C =$ collection of $C_1, C_2, \dots, C_j = \{C_1, C_2, \dots, C_j\}$

Algorithm a.2, refines clusters by splitting them based on their features (width-height ratio of a single bounding box, r , and area of the bounding box, s) using Frobenius norm to calculate their feature distance between any two boxes, Equation (3-6). This is because distance range alone is not enough to sufficiently split the clusters to correctly represent fish trials.

$$width\ height\ ratio = r_i = \frac{w_i}{h_i}$$

$$distance_2(i, j) = \sqrt{|r_2 - r_1|^2 + |s_2 - s_1|^2} \quad (3 - 6)$$

The reason why the Frobenius norm is used is because of its simplicity: the square root of the sum of the squared magnitude of all entries, which has the

same form as the Euclidean vector norm. If the distance of a bounding box center is greater than a set threshold (parameter a_4 , default 1000) from all the other bounding box centers, then that bounding box is omitted from further processing since it is most likely not going to help in further processing. If some boxes are further away from all boxes but their number is higher than a specific number (default threshold 3), then these boxes are considered a new cluster. The resulting clusters are further refined by splitting clusters where their bounding boxes do not intersect, using the concept of connected components. This is because clusters that do not intersect most likely represent 2 separate fish trials, according to preliminary observation during the development of this algorithm. Connected components mean if there are boxes whose pixels do not overlap with the other boxes within the same cluster, then those boxes will be segregated from the cluster and form another cluster. The boxes that have less than 3 points within the same cluster are omitted from further processes. Redundant boxes that take up the same space (IOU = 100%) are also omitted.

Algorithm a.2 (refinement of cluster)

Input: $C = \{C_1, C_2, \dots, C_{|C|}\}$

$j = 1$

Step 1

Calculate $\Delta(i, k) \equiv \text{distance}_2(i, k)$ for all pairs of $(xywh_i)$ and $(xywh_k) \in C_j$, and $i < k$

$SC_1 = \{(xywh_i), (xywh_k) \in C_j \mid \Delta(i, k) \leq a_4\}$

$SC_2 = C_j - SC_1$

If $(|SC_2| < 3)$, $C_j \leftarrow C_j - SC_2$, goto **Step 2** [Remove SC_2]

Else $C_{|C|+1} = SC_2$, $C \leftarrow C \cup \{C_{|C|+1}\}$, $C_j \leftarrow C_j - SC_2$ [Split C_j]

Step 2

If $(j < |C|)$, $j \leftarrow j + 1$ go to **Step 1**

$j = 1$

Step 3

Connected bounding boxes: Two boxes are said connected if they overlap. (Their intersection contains at least one pixel).

Partition C_j into l sets (clusters) of connected boxes, $\{P_1, \dots, P_l\}$.

(Any box in P_i is connected with, at least, another box in P_i .)

Any two boxes belonging to two separate clusters are not connected.)

If $(l \geq 2)$, $C_j = P_1, C_{|C|+1} = P_2, \dots, C_{|C|+l-1} = P_l$, $C \leftarrow C \cup \{C_{|C|+1}, \dots, C_{|C|+l-1}\}$

Else if $(j < |C|)$, $j \leftarrow j + 1$, goto **Step 3**

Remove any box in C_j that is completely contained (IOU = 100%) by another box in C_j .

Remove any cluster C_j in C if its size (number of boxes) is less than 3.

Renumber the cluster indices in C

Output: $C = \{C_1, C_2, \dots, C_{|C|}\}$

Algorithm a.3, shown below, refines clusters by examining the angle data. Not all angles are reliable; therefore, only a certain percentage (parameter a_5 , default 0.7) of the most similar angles are used. Similar angles are obtained by sorting the angle difference (in radians) of all pairs and the top $a_5\%$ are selected and their angles are used. Then, the standard deviation of those selected angles in a single cluster is calculated. If it is more than a specific threshold (parameter a_6 , default 0.1), it is omitted from further processes; otherwise, it goes to the next stage, **Algorithm a.4**. The reason for such a design is because objects such as corals usually produce angles that are erratic, as shown in Step a.3.2 in Figure.3.11. Often the mis-detected fish angles also have divergent

angles. Hence, omitting such clusters improves the final accuracy result. In a variation of our scheme, the original YOLO (not YOLO-Ang) is used to predict bounding boxes in a trail image; thus, the angle estimate associated with the box is not available. In this case, **Algorithm a.3** is skipped.

Algorithm a.3 (angle alignment)

Input: $C = \{C_1, C_2, \dots, C_{|C|}\}$

$j = 1$

Step 1

$\theta_i \equiv$ Estimated angle of $(xywh_i)$ [Angle estimated by YOLO-Ang]

Calculate angle difference $\Delta\theta(i, k)$

$= |\theta_i - \theta_k|$ for all pairs of $(xywh_i)$ and $(xywh_k) \in C_j$, with $i < k$

Sort $\{\Delta\theta(i, k)\}$ and select the top **a_5%** pairs with smallest $\Delta\theta(i, k)$ to form a set D

Calculate the standard deviation of all θ_i in $D \rightarrow \sigma(D)$

If $(\sigma(D) > \mathbf{a_6})$, $C \leftarrow C - C_j$ [Ignore this cluster C_j]

Else if $(j < |C|)$ $j \leftarrow j + 1$, goto **Step 1** [Keep this C_j]

Renumber the cluster indices in C

Output: $C = \{C_1, C_2, \dots, C_{|C|}\}$ [Clusters with consistent angles]

The purpose of **Algorithm a.4**, shown below, is to produce the final box that will be assigned as a *support box* for future processing. Essentially, a final support bounding box is derived from each cluster. Often, the current frame fish is at the two ends of a fish trail (trajectory). Therefore, we pick up the furthest two bounding boxes in a cluster. Then, we use the angle information derived from these two bounding box centers to find the final target box location. As shown in the original frame in Figure 3.12, there is a fish whose colour is very similar to the background. The detected trail is able to highlight the existence of a fish in that location. Therefore, it is very likely that the current frame has a fish at the end of the “trail”. In order to decide which direction of the trail the fish will be on, the angle information is used to predict which direction the fish

is heading. The final result is a support box that will be used in the next module, the Fusion module.

Algorithm a. 4 (final supporting box)

Input: $C = \{C_1, C_2, \dots, C_{|C|}\}$

Initial: $S = \phi$ [Support bounding box set; initial value = empty set]

$j = 1$

Step 1

Calculate $\delta(i, k) \equiv \text{distance}_1(i, k)$ for all pairs of $(xywh_i)$ and $(xywh_k) \in C_j, i < k$

Select the maximum $\{\delta(i, k)\}$ and call this pair $(xywh_1)$ and $(xywh_2)$

$\theta_{12} = \text{angle}((xywh_1), (xywh_2))$

$\theta_{21} = \text{angle}((xywh_2), (xywh_1))$

where $\text{angle}((xywh_A), (xywh_B)) \equiv \arctan(y_B - y_A, x_B - x_A)$

Step 2

$\bar{\theta}_D = \text{Average of all } \theta_i \text{ in set } D \text{ in } C_j; D \text{ was derived in Algorithm a. 3.}$

If $(|\theta_{12} - \bar{\theta}_D| > |\theta_{21} - \bar{\theta}_D|)$, final support box $b_j = (xywh_1)$

Else final support box $b_j = (xywh_2)$

$S \leftarrow S \cup \{b_j\}$

If $(j < |C|), j \leftarrow j + 1, \text{ goto Step 1.}$

Renumber the bounding box indices in S

Output: S

In the case where angle information is omitted in the clustering module, no refinement of clusters will occur. Angle data is also responsible for determining the final bounding box within a cluster. This is done by querying the furthest 2 points within a cluster, and then, depending on the angle direction, one of the 2 points is chosen as the final box. To replace this mechanism with the absence of angle, bounding boxes representing the current frame (which is class 0; if class 0 is absent, the lowest class is used) are used as candidates for the final support boxes. Since there are multiple class 0 boxes, boxes with the highest confidence are chosen as the final supporting box. Thus, the clustering module without angle is simpler than the original proposed clustering procedure.

The original proposed clustering and fusion module has a total of 8 parameters, while the without-angle procedures have 6 (a_5 and a_6 are omitted).

Figure 3.13 (left) shows a general bounding box. The integer in the box label represents class. The floating-point number in the label represents box confidence. Figure 3.13 (right) is a sample of a cluster. The box that is pointed by an arrow will be chosen as the final support box. It has the lowest class number (0), and also the highest confidence score among class 0.

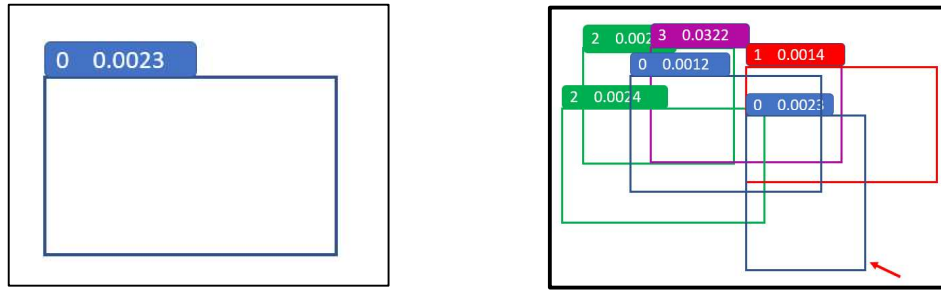


Figure 3.13: Sample of how the final supporting box is chosen in the case where angle data is omitted in clustering module

3.3.2 FUSION MODULE

Fusion module is a simple algorithm that helps merge support boxes with the low-confidence outputs generated by the original YOLOv5, as visualized in Figure 3.14. The fact that the support box exists suggests the possibility of a fish object being in that area. If it is supported with a low confidence threshold output from the original YOLOv5, then it is considered a detection. There are some steps and criteria that need to be met before low-confidence boxes are considered a correct detection.

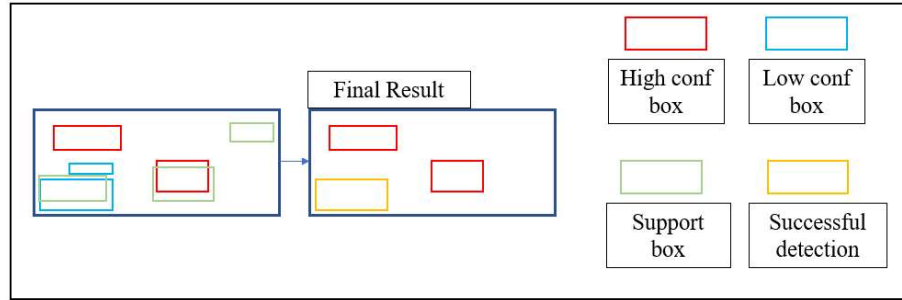


Figure 3.14: Fusion module flowchart: Low-confidence output that overlaps with the support boxes (satisfying certain criteria) is considered successful detection that eliminates FN errors.

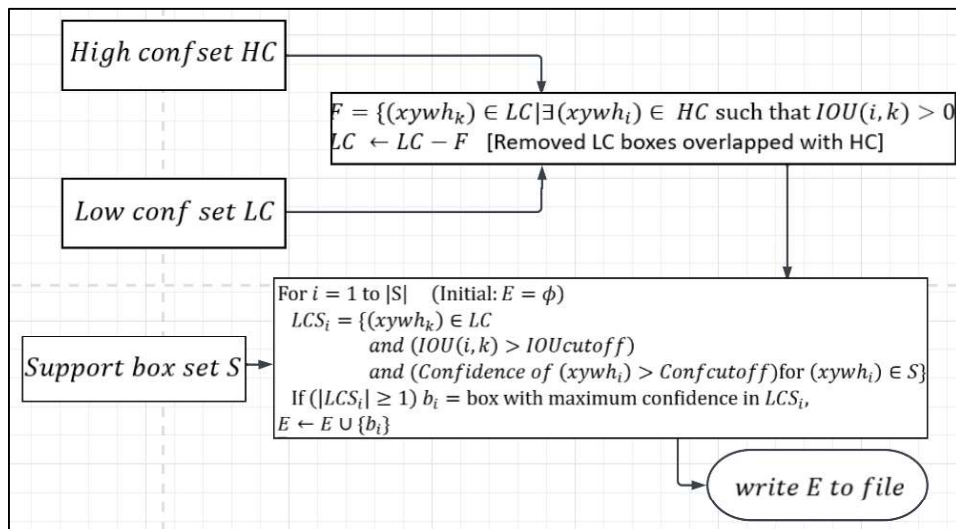


Figure 3.15: More detailed flowchart for fusion module

In Figure 3.15, let set HC be high confidence bounding boxes from YOLOv5 output. Let set LC be low confidence bounding boxes from YOLOv5 output. Let set S be the support boxes from the clustering module output. Let set E be the final extra box for the whole system. The confidence threshold is defined by the best validation confidence threshold when training the YOLOv5 model, which returns the best F1 score. In this case, it is 0.541. Bounding boxes lower than this threshold are considered low-confidence boxes (LC), and anything higher is considered high-confidence bounding boxes (HC). The

confidence lower bound for the low confidence boxes is set by the parameter *Confcutoff*, with the default value set at 0.001. First, the low-confidence boxes of YOLOv5 that overlap with high-confidence boxes, or do not overlap with support boxes, are omitted. Then, if there are multiple low confidence boxes overlapping with support boxes with $IOU > IOU_{cutoff}$ (default 0.1), the higher confidence box out of the multiple overlapped boxes is chosen and the rest are omitted. Finally, the corresponding low-confidence boxes that overlap with support boxes are considered valid detections.

3.3.3 DEFAULT PARAMETERS

There are a few adjustable parameters, and their default is set at: $a_1 = 0.05$, $a_2 = 0.5$, $a_3 = 20$, $a_4 = 1000$, $a_5 = 0.7$, $a_6 = 0.1$, $IOU_{cutoff} = 0.1$ and $Conf_{cutoff} = 0.001$.

The design of the cluster output and fusion module of the system is based on a few difficult image samples. Image difficulty is subjectively judged mainly based on misdetections from the original YOLOv5 output in the training set. The YOLOv5 model is trained on the training set, and detection is done on the training set again. The images where the model missed out on some fish instances are chosen as “hard samples”. Focusing on these frames, the proposed method is designed. The default parameters are manually adjusted, so previously undetected fish instances are detected.

The left row of Figure 3.16 is the sample images; the red bounding box represents the ground truth boxes. The right row pink bounding boxes represent high-confidence boxes, and the yellow boxes are the final support boxes generated by the proposed method.

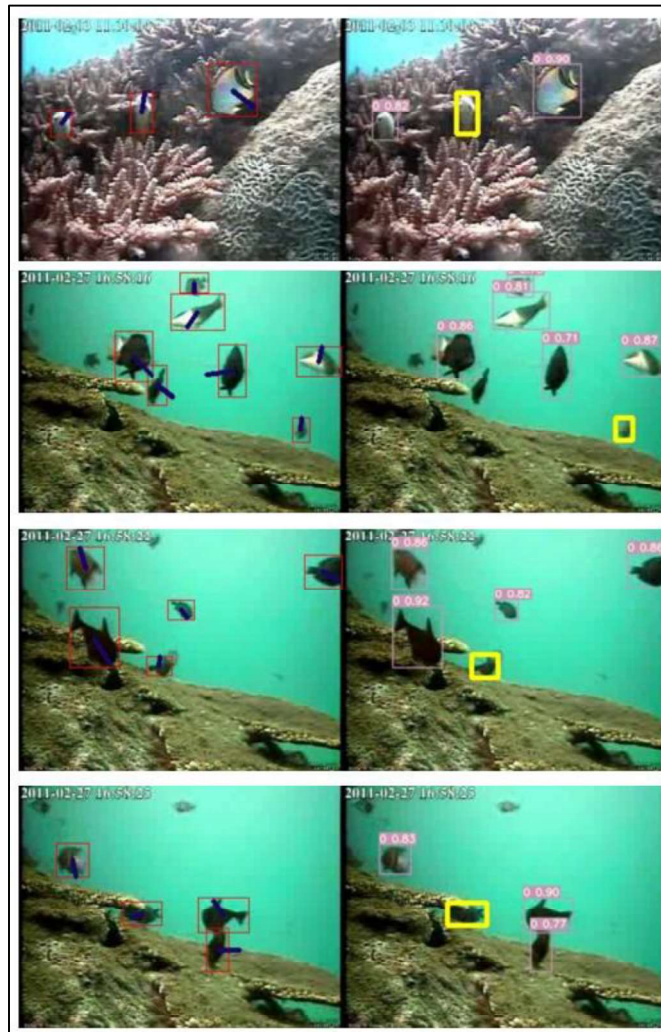


Figure 3.16: Successful cases picked up by the supporting boxes. Left row: ground truth. Right row: pink boxes represent high-confidence detections; yellow boxes represent final support boxes.

CHAPTER 4

ABLATION STUDY AND EVALUATIONS

The goal of the simulations in this chapter is to 1) confirm parameters set during the design of the proposed algorithm are optimum by running parameter optimization procedures, and 2) find out the effectiveness of the proposed method and its components.

We first introduce the dataset and its ground truth labels definition used in the experiments in Section 4.1. Experimental setup and brief metric explanation will be detailed in Section 4.2. YOLOv5 baseline experiment will be described in Section 4.3. Parameters from the post-processing modules will undergo parameter optimization in Section 4.4. Various system (component) variants are examined in the ablation studies in Sections 4.5 to 4.7. The simulations include running multiple tests on variations of the proposed method. This is to prove the effectiveness of some components and design choices in the proposed method. Specifically, Section 4.5 test out different trail formulation method and different number of frames to construct a trail image. In Section 4.6, we would test the effectiveness of angle data in both the YOLO-Ang and clustering module. Section 4.7 will do a brief test using MHI for the trail formulation method. Lastly, Section 4.8 describes the computational complexity of the proposed system.

4.1 FISH DATASET

The dataset used in this project is called FishCLEF-2015 [10]. It is the dataset used for the fish identification task in the LifeCLEF-2015 competition. The dataset consists of 93 videos, with bounding box annotations that represent species-constrained-fish locations. The training set of the dataset consists of a total of 9163 annotations, while the testing set contains 14199 annotations. The video resolution ranges from 320 x 240 to 640 x 480, with a frame rate of 25 fps.

The dataset also provides 20000 sample images of fish species. However, since this project does not do species classification, the sample images were not used. It is worth noting that temporal information is not exploited during the annotation process. This means annotators will not label a fish that is not clearly identifiable, despite the same fish being clearly identifiable in the previous frame. FishCLEF-2015 is a subset extracted from the data collected by the Fish4Knowledge project (Fisher, Shao and Chen, 2016), which records and analyses 90 thousand hours of video from camera locations on several tropical coral reefs off the coast of Taiwan.

This dataset has 15 fish species for the training set, and 23 species for their testing set. However, it did not define a specific validation set. We split the training data into training and validation set by 6:1.

The definition of ground-truth labels for training and testing is the same, and they include 15 species of fish, despite the fact that the testing set has 23 species. This is to make sure that our experiments conducted are fair, since the training set only has 15 species, therefore we only consider fishes within the 15 species as ground-truth in the testing set. For our experiments, we consider all 15 species as a single class, instead of classifying each species.

The challenges of underwater fish detection are described in many papers, such as [46]. These challenges include: 1) Low-resolution underwater fish video. Fish texture information may be lost. 2) Motion blur. For fast moving fishes, when there is insufficient light, it is difficult to detect whether it is a fish or other moving objects. 3) Complex seabed background, complicated coral reef backgrounds, and moving plants can cause detection algorithms to mistake some visual features. 4) Camouflage color. Fish objects may sometimes blend in with the background. 5) Occlusion. When the density of fish is high, the fish may overlap. Or the fish may simply be hiding behind rocks or other coral objects. 6) Small objects. Sometimes, the fish are far away and appear small. 7) Complex and varied underwater environment. Since the underwater data is collected at different times with different background scenes, there might be changes with different light intensities; this can be seen in Figure 4.1.

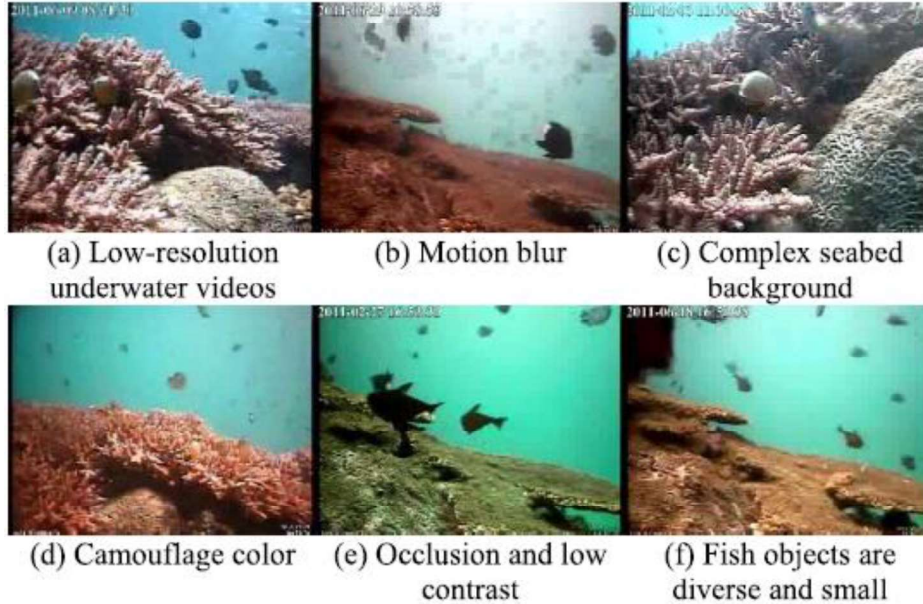


Figure 4.1: Six typical challenges that the FishCLEF-2015 dataset shows, extracted from [46].

The goal of the creator of this dataset (which was used for a challenge) is to detect fish belonging to 15 selected species. That is, a fish not belonging to one of these 15 species is not included in the ground truth labels. This would cause a serious problem if our goal is to detect the instance of any species of fish. For example, the binary YOLOv5 (trained for fish and non-fish only) detects quite a number of fish that do not belong to these 15 species.

One solution to this problem is using an extra class verifier that checks the fish class in the detected bounding box at the end of the detection process. It eliminates fish outside the specified 15 species. Another solution to the problem is modifying the ground truth labels to include fish of all kinds.

However, these aforementioned solutions are resource-intensive, and there is difficulty in determining which instances can be considered fish. Therefore, our proposed method is designed to perform well using the original ground-truth labels in the experiments.

4.2 EXPERIMENTAL SETUP

The metric used for the experiments is the F1 score. This is because our design includes a set confidence threshold and has an optimal operating point, where the F1 score is calculated. Whereas using mAP50, the results are computed across all possible confidence scores. This does not reflect the performance of the system, which was designed using a set confidence threshold (to separate low and high confidence boxes, and supporting boxes also have a set confidence value assigned).

The confidence threshold is defined by the best validation confidence threshold when training the YOLOv5 model, which returns the best F1 score. In this case, it is 0.541 (the experiment is explained in Section 4.2). Bounding boxes lower than this threshold is considered low confidence boxes, and anything higher is considered high confidence bounding boxes. A detection is considered successful when the IOU of the prediction box and ground truth is more than 0.5.

The simulations are run with Intel® Xeon® CPU E5-2620 v5 @ 2.10GHz. The experiment models for both the original YOLOv5 and YOLO-Ang are trained on the NVIDIA GeForce GTX 1080 GPU, which has 2560 CUDA Cores, CUDA version 10.1, 1607 MHz Graphics Clock, 1733 Processor Clock and 10 Gbps Memory Clock. Stochastic gradient descent with an Adam optimizer with a learning rate of 0.01 is used, with a momentum of 0.937 and 0.0005 weight decay. 3 warmup epochs are used with 0.8 warmup momentum. The augmentations used are the default setup from the authors of YOLOv5, with 0.015, 0.7, 0.4, 0.1, 0.5, 0.5, 1.0 probability for hue, saturation, value, translation, scale, left-right flip and mosaic augmentation, respectively. All the models are trained on 4 images per batch, with a pretrained YOLOv5 model that was trained on the ImageNet dataset. The pretrained model can be found in [12].

4.3 YOLOV5 BASELINE EXPERIMENT

We trained a baseline model using the original YOLOv5. Figure 4.2 shows the learning curve for training data and Figure 4.3 for validation data. These learning curves are to simply show that the model is converging.

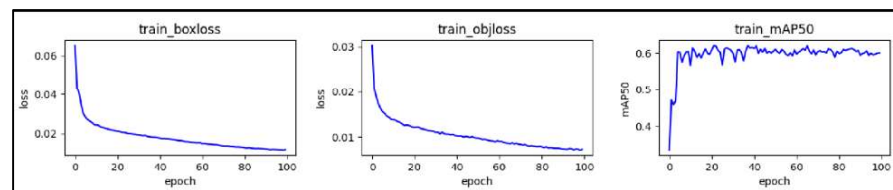


Figure 4.2: Learning curves for training data

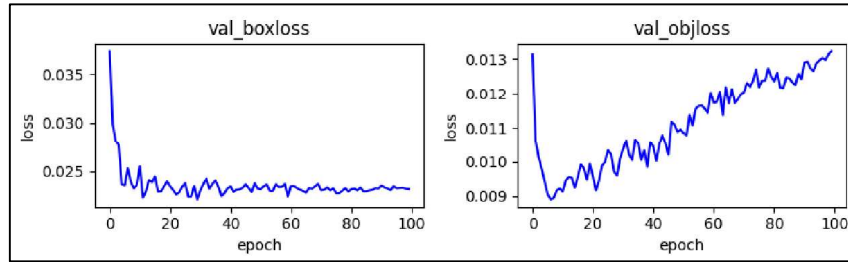


Figure 4.3: Learning curves for validation data

The learning curve for the YOLOv5 baseline experiment seems reasonable, indicating the model is learning properly. The best loss for validation data was achieved at the 11th epoch. According to how validation loss increases around the 11th epoch, it can be said that the training converges around the 11th epoch. After the 11th epoch, there are signs of overfitting as the training loss keeps decreasing for validation data while object loss keeps increasing for validation data. The box loss calculates the IOU of only the prediction boxes that fall within the target box index; the rest of the boxes are ignored. The objectness loss *predicts* the IOU of *all* prediction boxes. Observing the different learning curve behaviors of validation box loss and validation objectness loss, it seems like the model is able to encapsulate the fish object well (if the prediction box is within the target index) despite having an overfitting issue at later epochs.

The result of the YOLOv5 baseline experiment is as follow. The F1 score for training data is **96%** at its best confidence of 0.532 (TP: 7921, FN: 245, FP: 339), and for validation, it is **85%** at its best confidence of 0.541 (TP: 832, FN: 155, FP: 136). For testing, a validation best confidence of 0.541 is used, and the model was able to reach a **59.02%** F1 score (TP: 6574, FN: 5115, FP: 4013).

4.4 PARAMETER OPTIMIZATION

There are a few adjustable parameters in the post-processing module. The parameters are set in a way that produces the best result while designing the algorithm with a few hard sample frames, as mentioned in Section 3.3.3. In order to make sure these parameters are truly optimal, optimization is conducted.

There are several parameter optimization methods [84]; three will be explained: grid search, random search, and Bayesian optimization. The first and one of the most traditional ways is grid search. Grid search is an exhaustive search that goes through the parameter space. Grid search will ultimately reach the optimum parameters. However, it suffers from the curse of dimensionality. Since there are a total of a combination of 8 parameters, with unbounded value spaces for some parameters, it is computationally unreasonable to perform a grid search.

Random search replaces exhaustive search with grid search by selecting the parameters randomly. It can computationally outperform grid search; however, it risks falling into a local minimum. Despite its simplicity, random search is generally used as a baseline to compare the performance of new parameter optimization methods.

Bayesian optimization iteratively evaluates parameter combinations and performs updates to those parameters so that the parameter configuration can achieve the best value for an algorithm according to an objective function.

According to the papers discussing optimization procedures [85], [86], [87], and [88], Bayesian optimization achieves better results in fewer iterations compared to grid and random searches. This is because Bayesian optimization has the ability to evaluate the quality of experiments before going to the next iteration. Therefore, Bayesian optimization is used for the parameter optimization. Particularly, we use the hyperopt library [89], using the F1 score as the objective function.

Here, in Table 4.1, we describe the adjustable parameters used, their description, their default values, their range, and the sampling space. Note that the sampling space uses a normally-distributed range with a mean and standard deviation. The mean uses the default values, and the standard deviation is shown in the table. For the clustering module without angle, the parameters are the same except for parameters a_5 and a_6 , which are related to the angle information and are omitted.

Table 4.1: Parameter values used for clustering and fusion modules

Parameter	Description	Default values/mean	Range	Standard deviation
a_1	Threshold to remove small boxes	0.05	0 - 1	0.001
a_2	Threshold to remove big boxes	0.5	0 - 1	0.1
a_3	Distance threshold between 2 boxes	20	No limit	10
a_4	Linalg distance between boxes within a cluster	1000	No limit	100
a_5	Threshold for the most similar angle	0.7	0 - 1	0.1
a_6	Standard deviation accepted for group of angle	0.1	0 - 1	0.05
IOU_cutoff	IOU cutoff threshold	0.1	0 - 1	0.025
Conf_cutoff	Confidence cutoff threshold	0.001	0 - 1	0.0001

In order to make a fair assessment by avoiding optimizing the testing data, the optimization algorithm optimizes the parameters in the training and validation data. Firstly, training and validation data are processed using the trained YOLOv5 and YOLO-Ang models. The outputs are then stored. Then, with a set of initial parameters, we adjust the parameters by applying the clustering and fusion modules to the stored output data. The F1 scores are then calculated based on the final output of the system. Each iteration of the optimization procedures goes through the whole system, from YOLOv5, YOLO-Angle, clustering, and fusion modules, and calculates the F1 score of the final output. The parameters for the first iteration use the default parameters. Then, during the next iteration, the optimization algorithm will update the parameters according to the previous iterations' F1 score, with the goal of improving the F1 score. We set the maximum number of iterations to 300.

In order to avoid reaching a local optimum, the parameters are periodically deviated from the usual update direction for certain iterations. If no improvements are made, the updates will continue in the usual direction.

The optimized parameters belong to the iteration with the highest F1 score. Then, the whole proposed system will use the optimized parameters to perform tests on the testing data. The results of the tests are shown in Table 4.2. Parameter optimizations are done on the most basic setting, which is *5f5bb (trail formulation using 5 frames and bounding boxes from 5 frames)*, with trail addition using YOLO-Ang trained with angle, and with clustering module using angle.

Table 4.2: Optimized parameter performance: trail image with addition method, trained with angle data, cluster module with angle data

model	a_1	a_2	a_3	a_4	a_5	a_6	IOU_ cutoff	conf_cut off	F1 (improvement) 15 species
5f5bb	0.050	0.37	0.00	1123.19	0.60	0.01	0.09	0.0009	59.02 (+0)

It seems there were barely any improvements. This might be because optimization procedures are done to set the parameters of the clustering and fusion modules.

Optimizing the training and validation data would mean optimizing the output of the YOLOv5 model on the training and validation data. Since the model is trained on the training data, its original YOLOv5 output's F1 score already reached 96% on training data and 85% for validation data, which leaves little space for further improvements.

Therefore, the assumption is that when the addition support box is introduced, it is unlikely to be an extra True Positive and most likely to introduce a False Positive which brings down the F1 score. The optimization procedure then sets the parameters in such a way that no additional support boxes are introduced to preserve the already high F1 score for the training and validation data. At the end of optimization, the auxiliary system parameter values are selected to produce no extra support bounding boxes, which practically disables the functionality of the auxiliary system. Therefore, clearly, we can predict that this optimization procedure is not able to produce "improvement" since no extra support boxes are generated.

Since optimizing parameters results in almost no improvements, we use the default parameters throughout the rest of the experiments. Since the default parameters were manually set based on a few hard samples. It can be said that the optimization process proved that the manually set parameters performed best.

4.5 TRAIL IMAGE FORMULATION AND YOLO-ANG CONFIGURATION

There are 2 types of trail image formulation, as explained in Sections 3.1.1.1 and 3.1.1.2. The first type is to use the addition method to merge frame differences into a single frame, and the second type is to use the overlapping method. For each type of trail image formulation, there are 2 types of configurations of YOLO-Ang. It can be specified by two numbers denoted by f , the number of frames to construct trail images, and bb , the number of bounding boxes' frames. The first number is the number of frames that make up a trail image in the trail formulation module. The second number is the number of frames that contain the predicted bounding boxes (objects). Let frame #1 be the current frame, and frame #4 be the frame 3 frames before the current frame. For example, frames #1 to #4 are used to construct a trail image, but the detector predicts the bounding boxes belonging to only frames #1 to #3. Thus, in this example, the first number is 4, and the second number is 3, $4/3bb$. The first configuration of YOLO-Ang is $n/f(n-1)bb$, and the second is $nfnbb$. The rest of the parameters used in the auxiliary system are at their default values, which were set during the design of the algorithm. The improvement refers to the

percentage improvement from the original YOLOv5.

Figures 4.4 and 4.5 show the learning curves of training and validation data, respectively, for model *4f4bb*, while Figures 4.6 and 4.7 show the learning curves of training and validation data, respectively, for model *5f5bb*, and Figures 4.8 and 4.9 show the learning curves of training and validation data, respectively, for model *6f6bb*.

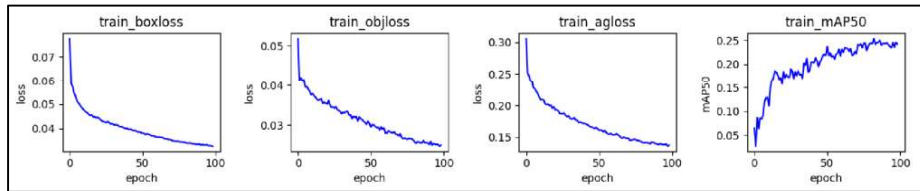


Figure 4.4: Learning curves on training data for model *4f4bb*

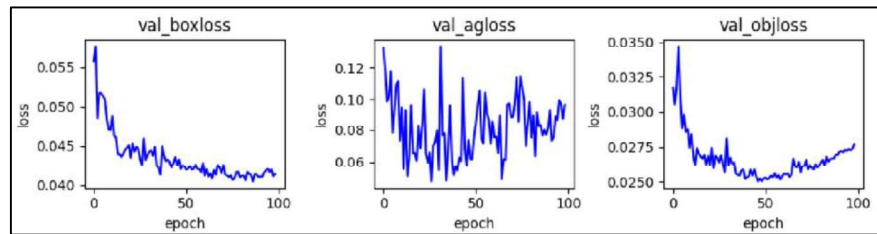


Figure 4.5: Learning curve on validation data for model *4f4bb*

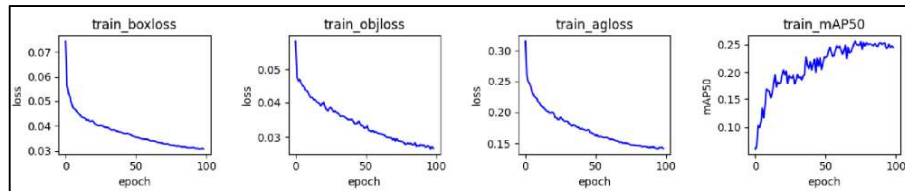


Figure 4.6: Learning curves on training data for model *5f5bb*

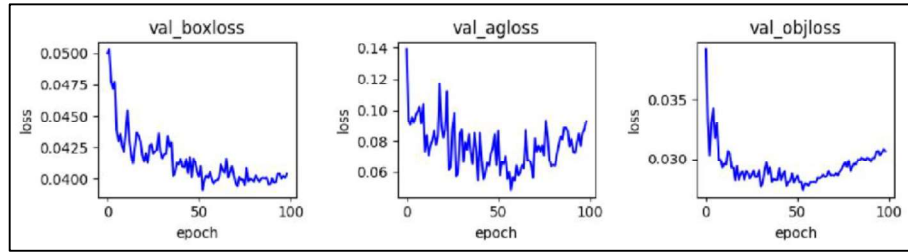


Figure 4.7: Learning curves on validation data for model 5f5bb

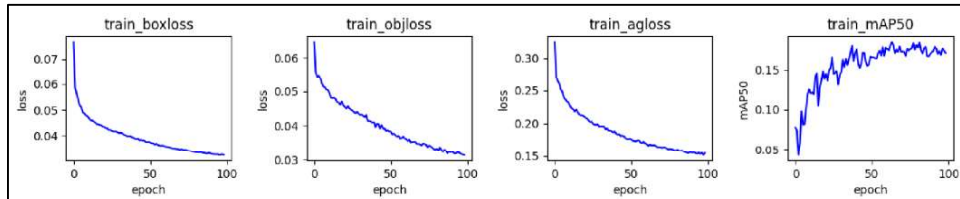


Figure 4.8: Learning curves on training data for model 6f6bb

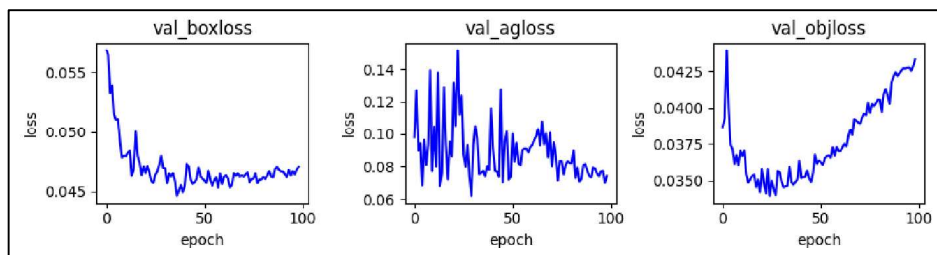


Figure 4.9: Learning curves on validation data for model 6f6bb

Some insights that can be derived from the learning curve are that, the models are trained correctly, which means they are learning properly, with loss values gradually decreasing after every epoch in the training phase. It can also be observed that model overfitting usually occurs after the 50th epoch (except for 6f6bb, where it happens at epoch 29) as loss values start to increase in the validation set.

The best loss for validation data reached for model 4f4bb is at epoch 64; for model 5f5bb, it is at epoch 57; for model 6f6bb, it is even lower at epoch 29.

According to the training data for all the models, the loss for all models gradually lowers; however, overfitting might occur as the validation curve saw a gradual increase in object-ness loss after the best loss epoch, and as for angle loss, after the initial decline in loss, the decline is not stable with frequent spikes. For box loss, the decline is more consistent. For each model using *4f4bb*, *5f5bb*, and *6f6bb*, the F1 score for training data is 87%, 87%, and 82%, respectively; however, for validation data, there is quite a gap, at 63%, 64%, and 54%, respectively.

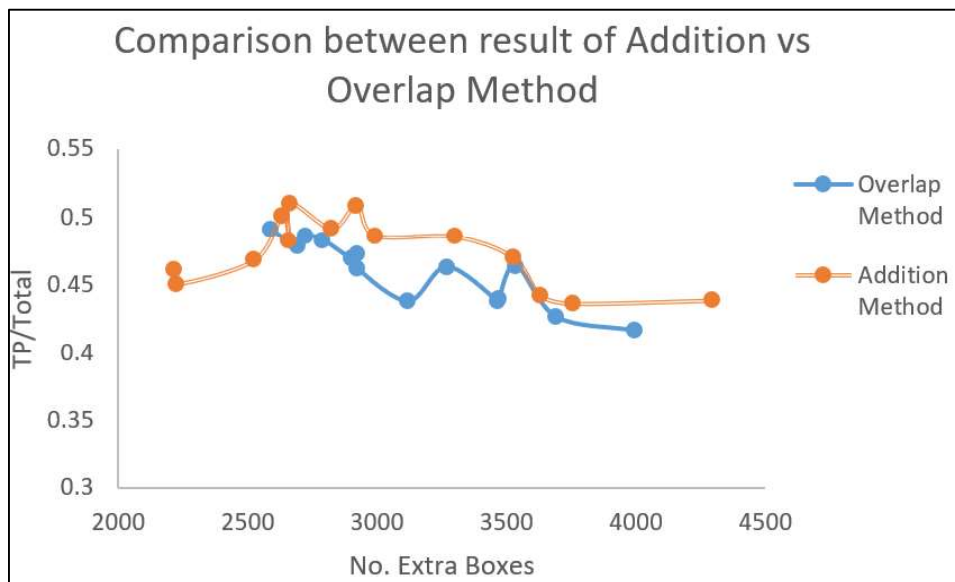


Figure 4.10: Comparison between the addition and overlap methods

The figure above (Figure 4.10) are trends derived from the ablation study between the addition and overlap trail formulation methods. The raw data can be found in the Appendix chapter. The above figure's y-axis shows the ratio between extra TP boxes and total extra boxes generated by the auxiliary system. Naturally, a higher value indicates a better F1 score. The x-axis shows the

number of extra boxes generated.

From the figure, it is easy to see that the addition trail formulation method consistently performs better than the overlap trail formulation method. A hypothesis of such phenomena is that, despite addition suffering from mixed signals, the neural network was able to cope with them. An insight into this observation is that “subjective neatness” (overlap image trail formulation results are “cleaner” subjectively) does not translate to better CNN performance.

We can conclude that overall, despite the number of FP cases increasing, the overall TP cases increase as well, and the overall FN cases decrease. Therefore, in the end, the overall F1 score increases. In the *5f5bb* trail image with the addition method, the auxiliary system reduces 1485 FN samples, which achieves the target of our design. That is, the auxiliary system provides an additional 1485 TP samples, although it also introduces an additional 1435 FP samples. Thus, the number of total TP samples is increased to 8059, the total FP samples are increased to 5448, and the total FN samples are reduced to 3660. In the end, the F1 score is 63.97%, 4.95% higher than the original YOLOV5 without the proposed auxiliary system.

However, the more boxes generated does not translate to a better TP over the total box ratio. If there are more than 3.5k extra boxes generated, the TP over total box ratio starts to decline rapidly for both trail formulation methods.

The highest F1 score improvement is at 4.95% with model *5f5bb* using the addition method. Therefore, the addition method seems to give better performance for default parameters, although the overlap method is not far off, with a difference in improvement scores of about 0.3%. It is interesting to see that the F1 score has a larger variation for the addition trail images compared to the overlap trail images. In summary, Table 4.3 shows a summary of the best-performing model for addition and overlap methods.

Table 4.3: Summary of best-performing models for different trail formulation methods.

Trail Formulation Method	Best model	F1 score (improvement)	Extra TP	Extra FP
Addition	5f5bb	63.97 (+4.95)	1485	1435
Overlap	5f5bb	63.66 (+4.64)	1642	1894

4.6 OMITTING ANGLE INFORMATION IN YOLO-ANG AND CLUSTERING

Angle information is used in 2 parts of the proposed scheme. One is in training the YOLO model; another one is in the clustering module. In order to test the contribution of angle information to the improvement of overall accuracy, angle information is omitted in each of the above two parts, and the results are compared with when it is included.

Essentially, we have two types of YOLO models for two types of auxiliary systems that reduce false positives. For the first model, we construct ground-truth with the angle information to train the proposed YOLO-Angle.

YOLO-Angle outputs the predicted angles in the inference phase. For the second model, the ground-truth does not include the angle information in training; that is, we use the trail images to train the original YOLOv5 architecture with multiple classes (classes are the frame indexes of bounding boxes) instead of the proposed YOLO-Angle. The output of the second model does not include any angle information. In the inference phase, both models accept the same trail image as input.

Two types of clustering module variations are designed to test the contribution of angle information. The first one is a combination of a no-angle YOLO detector with a no-angle clustering module. The second is YOLO-Angle with a no-angle clustering module. It is not possible to test the no-angle YOLO detector together with the with-angle clustering module. Raw data of the comparison of the original proposed method and the above two variations, which include the angle in the YOLO detector and clustering module, with both $n/n-1$ and n/n trail image formulations, respectively, can be found in the Appendix. As explained before, the total value of TP and FP adds extra TP and extra FP (from the auxiliary system) to the results from the original YOLOv5 RGB output. While the total FN is the original YOLOv5 RGB output FN, subtract the extra TP produced by the auxiliary system.

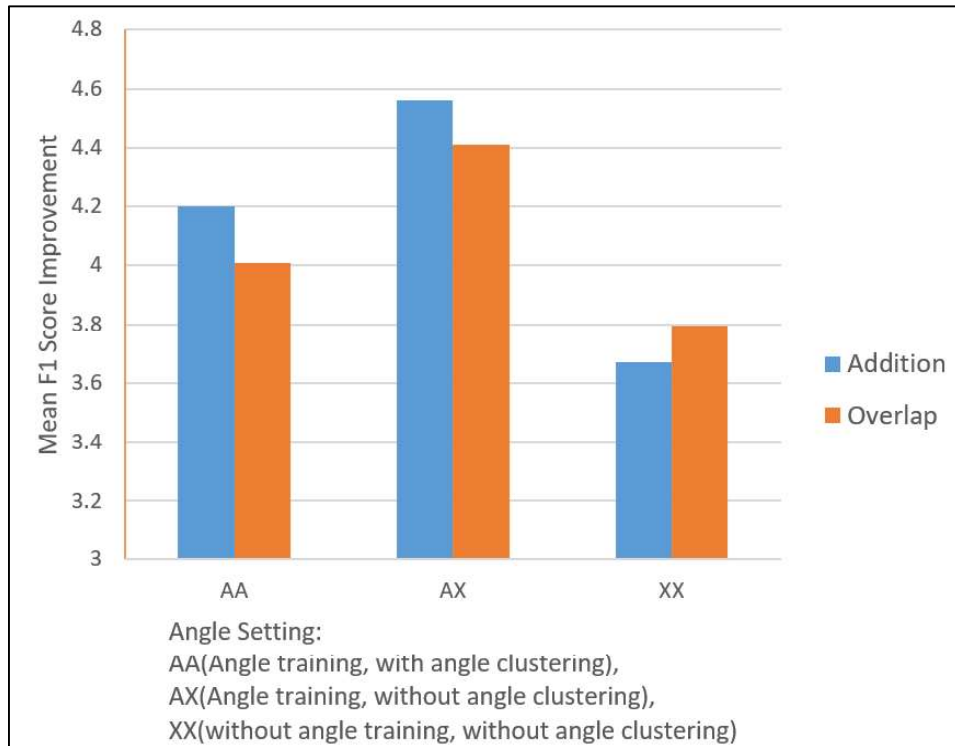


Figure 4.11: Comparison between different angle settings

Figure 4.11 shows a bar chart that compare different angle settings with the mean F1 score achieved. Essentially, all ablation results using the same angle setting are averaged to produce the final mean F1 score. For example, F1 scores of ablation studies from 7 different numbers of frames (3f-9f), with 2 types of numbers of bounding boxes ($nfnbb$ and $nf(n-1)bb$), from a single trail formulation method (exp: Addition), are added together and divided by 14, to produce a single bar.

From observation, generally, when there is no angle involved in both the YOLO detector and clustering module, there is the least amount of F1 score improvement compared to the other 2 settings that have angle involvement. However, when YOLO-Angle replaces the ordinary YOLO to detect train trail

images, but the output angle information is not used in the clustering module, there is a slight improvement in accuracy compared to using angle for **both** modules. One possible reason as to why omitting angle information in the clustering module produces better result is that: choosing a potential box from a cluster based on confidence scores is more reliable than picking 2 boxes that are furthest away from each other within a cluster, and picking the box based on angle direction. Therefore, the best setting is to use angle information while training YOLO-Angle and omit angle information in the clustering module.

The figures below (Figure 4.12, Figure 4.13) are graphs of different models using different trail formulation methods under default parameters. The y axis represents the value of F1 score improvement, and the x axis represents the number of frames used to construct a single trail image. The brackets within each series in the legend show if angle information is used during trail image training and also if angle is used in the clustering module. “withag” means angle is used during the training of trail images, while “NOag” means angle is not used. “withagcl” means angle is used in the clustering module, while “NOagcl” means it is not used.

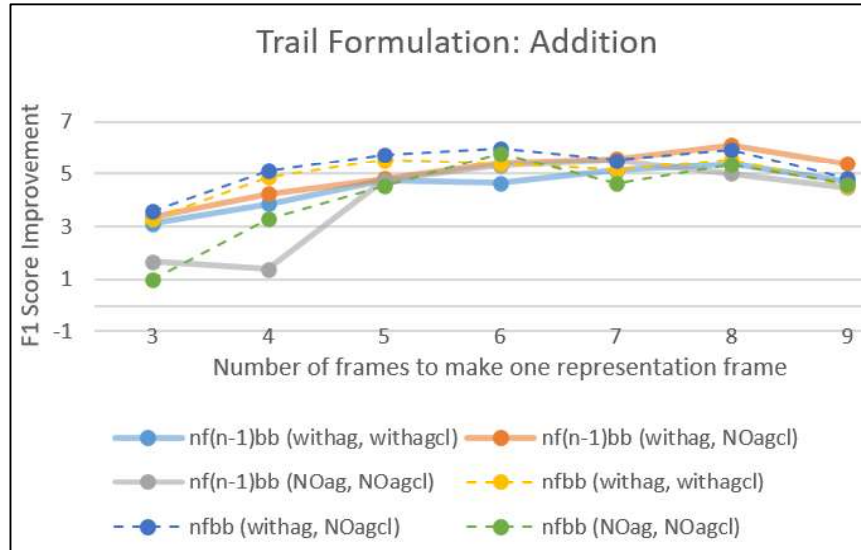


Figure 4.12: Results of different models under default parameters for the addition method.

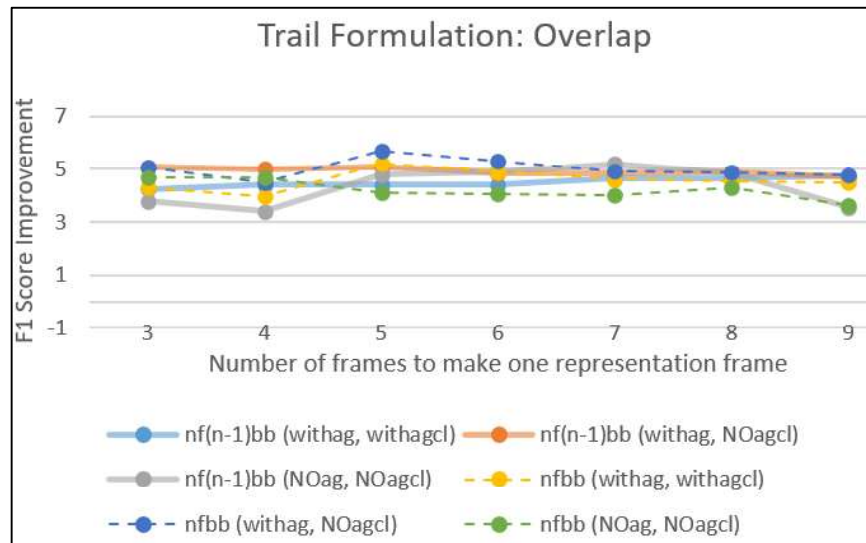


Figure 4.13: Results of different models under default parameters for the overlap method.

The above 2 line-graphs plots the relationship between the number of frames and the improvement of the F1 score at different angle settings. It shows that the absence of angle information seems to amplify the effects of f and bb , where the difference of extra bounding boxes between $3f$ and $5f$ is the largest

when there is no angle information involved in both components for the addition method; it is less obvious in the overlap method. Also, the peak performance moves to slightly higher frame numbers in trails. For trails formed by the addition method, the best F1 scores appear at *8f7bb* and *6f6bb*.

Angle information in training might affect the detection accuracy of trail images; therefore, some visualization, as shown in Figure 4.14, is presented to see the difference in the output between with angle training and without angle training for detecting the trail images. Generally, the settings that achieve the best result are trail formulation using the addition method and *nfnbb* trained with YOLO-Ang, where the clustering module does not use angle. The best number of frames that achieve the highest F1 score improvement is *6f6bb* at **5.28%**. It is also observed that using trail formulation training with angle, the clustering module without angle works better than the one with angle information. Also, the results start deteriorating around *7f*, where results for *9f* consistently have obvious deterioration for all types of settings. The results also show that the auxiliary system is able to detect difficult samples of fish, that the original YOLOv5 was not able to detect.

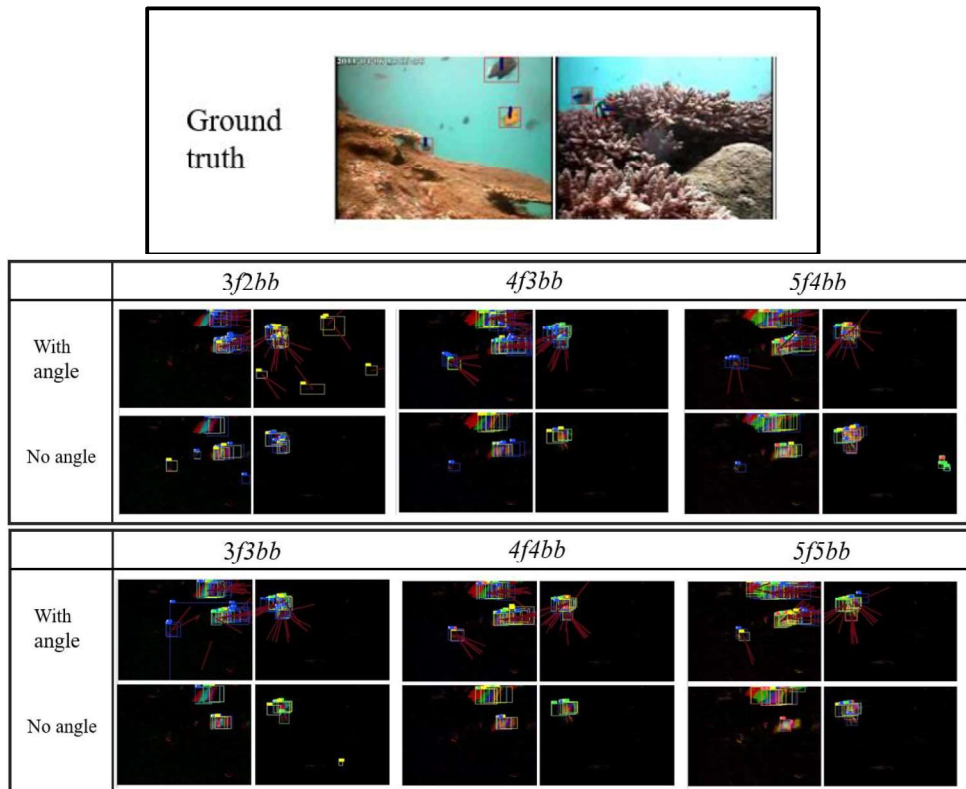


Figure 4.14: Difference in output between with and without training with angle data for trail images

Figure 4.14 shows that, subjectively, the trained trail image detector with angle is able to detect less obvious frame differences (pixels with lower brightness). This might be the reason why training with angle trail images is able to produce more supporting boxes; thus, increasing the chances of overlapping with a correct low confidence box, which leads to an increase in TP and ultimately, F1 score.

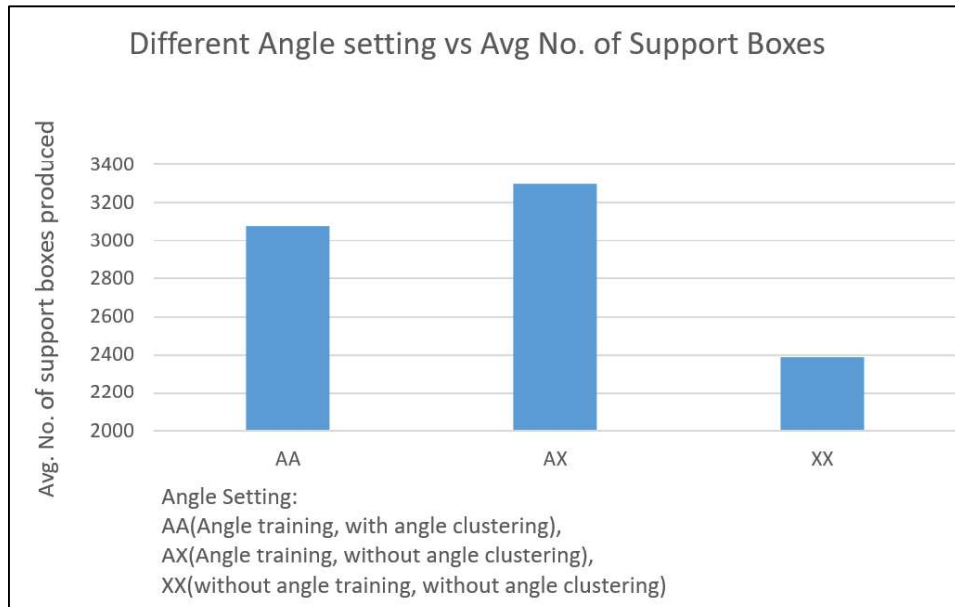


Figure 4.15. Relationship between different angle settings and the average number of support boxes produced

Figure 4.15 above shows that fewer supporting bounding boxes are generated when the YOLO detector is trained without an angle ground-truth. Another possible reason is that simply having extra information enables the model to learn better features. Ultimately, through the results shown in later subsections, despite not having a subjectively significant difference between training with and without angle, the observation that performance is better using images trained with angle is consistent.

4.7 REPLACING TRAIL FORMULATION BY MHI

The main difference between general MHI and the proposed trail image formulation module can be explained in the HSV color space. For MHI, the frame differences are expressed in a one-channel grayscale; thus, only the V channel is providing information. Time encoding is represented in terms of a decay function, which is updated solely on the V channel. To further explain, the V channel provides both time information and the location of movement. The MHI, expressed as $H_{\tau}(x, y, t)$, defined by equation (4-1) (which is based on [16]), where (x, y) and t represents position and time $\Psi(x, y, t)$ is the update function; it represents the moving pixels of an object in the current frame. τ represents the temporal extent, which can also be explained as the number of frames used to form a single MHI, and δ is the decay parameter:

$$H_{\tau}(x, y, t) = \begin{cases} 255 & \text{if } \Psi(x, y, t) = 1 \\ \max(0, H_{\tau}(x, y, t - 1) - \delta) & \text{otherwise} \end{cases} \quad (4-1)$$

To begin, the frame differences of the whole temporal extent of RGB frames are converted to grayscale. Then, the OTSU threshold is used to binarize the grayscale images. Then, the decay parameter is subtracted from the pixels of the binarized image ($\delta = \frac{255}{\tau}$). The result is a one-channel MHI frame, where the brighter pixels are the more recently moving pixels and vice versa.

For the trail image formulation module, since there is no binarization, both H and V channels are providing information; time encoding is represented as a change in hue values. To further explain, the H channel provides time information, while the V channel provides both the location of movement and the magnitude of the difference of a single pixel, which can be interpreted as preserving the representation of fish texture. In short, trail image formulation provides 3 types of information, whereas MHI only provides 2 types. Table 4.4 below summarizes the differences.

Table 4.4: Summary of the differences between trail image formulation and general MHI

Method	Types of information
Trail image formulation [proposed method]	Time (H), movement location (V), magnitude of pixel difference (V)
General MHI	Time (V), movement location (V)

In order to test the effectiveness of providing intensity of movement information, the MHI method replaces the trail formulation module in this experiment. Figure 4.16 illustrates the creation of MHI. The OTSU binarization method is chosen due to its ability to dynamically set a different threshold depending on the conditions of the frame. This is useful for offshore underwater conditions, which have constant changes throughout the whole frame. Since MHI only has one channel, the first layer of YOLO-Ang can be changed to accept one channel input, thus lowering computational complexity. Steps to generate MHI used by the auxiliary system are shown in Figure 4.16. In this figure, the binarized values of the support function (image) are 0 and 255 (“255” indicates value “1” in binarization) for display purposes.

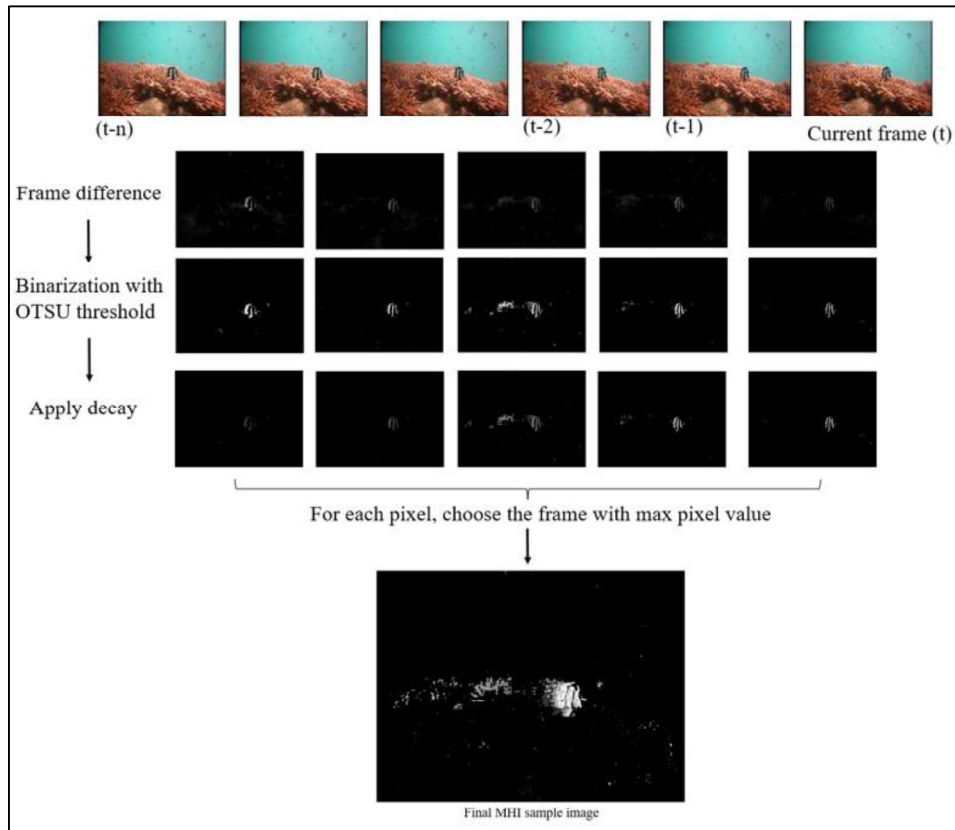


Figure 4.16: Flowchart and visualization of constructing an MHI image

The experiments that test the effectiveness of angle information are also tested with the MHI method to make a fair comparison. Raw results of MHI ablation studies can be found in the Appendix chapter. A line graph is plotted (Figure 4.17) to summarize the results of the trail formulation method using MHI.

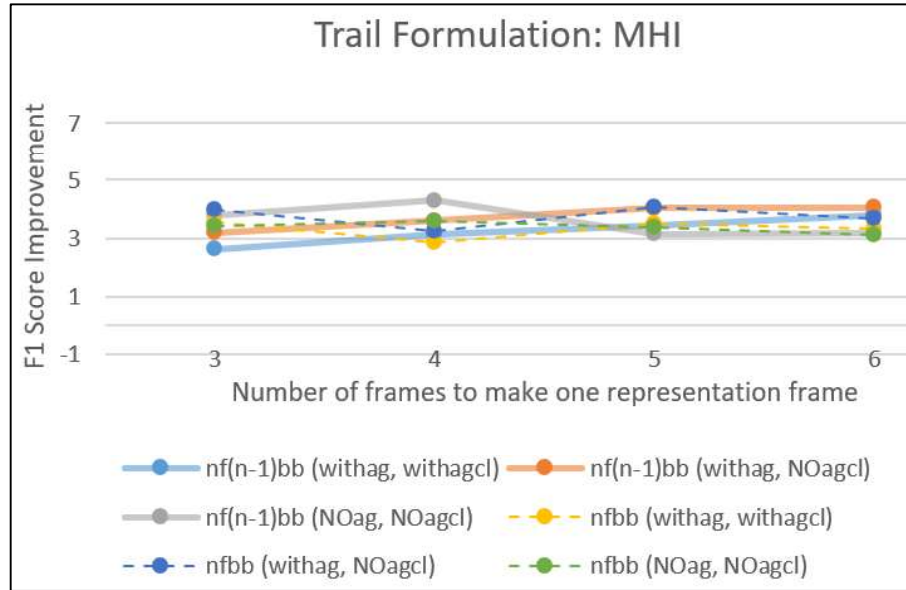


Figure 4.17: Results of different models under default parameters for the MHI method

It seems that generally, using MHI lowers the improvement on the F1 score. The accuracy starts deteriorating around 5f. The maximum F1 score improvement using MHI under the default parameter values is around 3.78%, which is 1.5% lower than the proposed trail image formulation using the addition method. Using MHI seems to increase the number of FP to the point where they are greater than TP, generating up to 3696, as compared to 2560 for trail image formulation using the overlap method. Table 4.5 shows the summary of the best-performing models of different trail formulation methods and their corresponding configurations.

Table 4.5: Summary of best-performing models of different trail formulation methods and their corresponding configurations

Trail formulation method	Best model	Trail image angle status	Cluster module angle status	F1 score (improvement)	CS $e^{((TP+FP)-NgI/NgI)}$	NCS (CS x Pr)
Addition	6f6bb	Yes	No	64.3 (+5.28)	0.561	0.393
Overlap	5f5bb	Yes	No	64.09 (+5.07)	0.567	0.397
MHI	4f3bb	No	No	62.61 (+3.59)	0.542	0.369

Table 4.6 shows a summary of other fish detection models' performance. These works were selected because the dataset and performance metrics used are similar to ours (LCF-15 and F1 score). There are other fish detection models that are not mentioned here and have been summarized in Table 2.2 in Section 2.3.5.

Similar to our proposed method, most state-of-the-art fish detection models are developed on existing baseline models. A comparison is made using the amount of improvement each state-of-the-art method achieved in relation to their respective baseline model. As can be seen in the table below, our proposed method achieved the highest amount of F1 score improvement at 5.28%.

Table 4.6: Selected works for comparison

Method	Standard baseline model	Dataset and description	F1 Accuracy	Baseline Accuracy	Amount of improvement
Hybrid System [28]	R-CNN	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	80.02%	77.30%	2.72%
Faster R-CNN + Shared RPN fusion [27]	Faster R-CNN	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	80.22%	78.78%	1.44%
YOLO + GMM [26]	YOLO	LCF-15 [with extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	95.47%	90.67%	4.80%
Our proposed method	YOLOv5	LCF-15 [without extra ROI images included] <ul style="list-style-type: none"> • Complex background • Blurry scenes • Varying illumination 	64.30%	59.02%	5.28%%

4.8 COMPUTATIONAL COMPLEXITY

A set of randomly picked sequences of 50 frames is used to calculate a comparison of the time-space complexity of different variations; it is recorded in Tables 4.7 to 4.9. The *5f5bb* frame setting for trail image construction is used throughout all methods for a fair comparison. Using the “tracemalloc” library for allocated memory size and the “timeit” library for run time measurement.

There is a trade-off between computational cost and accuracy improvement. Further work can be done to optimize code to reduce the trade-off gap. We use the library thops by L.Zhu [90] to compute the GFLOPs of the model. Note that for every YOLOv5 version, the GLOPs vary slightly; for example, in the v3 commit [91], the GFLOPs are at 39.4, while for another version in v4 [92], the GFLOP is at 51.3. The commit that is used in this project is within the many commits in v4 [93], and the GLOP is at 51.3. Note that the trail formulation module, clustering module, and fusion module are all run on the CPU, while variants of the YOLO model are run on the GPU.

Table 4.7: Computational complexity of different variations for trail formulation methods

Trail Formulation Method (on CPU)	Addition	Overlap	MHI
Time (average of 50)	0.005	0.008	0.002
Memory size (MB)	4.7	13.9	3.08

Table 4.8: Computational complexity of YOLOv5 and proposed variants

YOLO Type (on GPU)	YOLOv5	YOLOv5	YOLO_Ang	YOLOv5_single channel (For MHI)	YOLO_Ang_single channel (for MHI)
	<i>For RGB</i>	<i>For trail image</i>			
No. of parameters (M)	21.0	21.0	21.1	21.0	21.0
GFLOPS	51.3	51.3	51.5	49.6	49.8
Memory size (MB)	737	802	802	739	739
Inference time (average of 50)	0.021 (s)	0.023 (s)	0.024 (s)	0.022(s)	0.025(s)

Table 4.9: Computational complexity of different clustering and fusion methods

Clustering type (on CPU)	With angle	Without angle
Inference time (average of 50)	0.020	0.016
Memory size (MB)	0.2	0.5
Fusion (on CPU)		
Inference time (average of 50)	0.004	
Memory size (MB)	0.2	

The proposed system introduces some additional complexity and an increase in inference time, with YOLO_Ang, trail formulation using the overlap method, and with-angle clustering time taking up the most resources. The best-performing model, which is to use YOLO_Ang with addition trail formulation, and without-angle clustering, can reach up to 20 fps (frames per second). While the original YOLOv5 can reach up to 47 fps. The best-performing model can offer an F1 score improvement of 5.28% compared with the original YOLOv5. Future work may further improve efficiency.

Figure 4.18 shows some samples of successful cases where the auxiliary system helps detect fish that the original YOLOv5 fails to detect. We use the best model, which is the trail image with addition method, using angle training without an angle clustering module, with a *6f6bb* model with default parameters.

The first column shows the ground truth, the second column shows the trail image bounding boxes with different colored boxes representing different frames' boxes, the third column shows the clustering result after the clustering module, and the last column of blue bounding boxes shows the original YOLOv5 high confidence boxes, while the yellow box shows the supporting boxes that were produced by the auxiliary system. As can be seen in the figure, the auxiliary system is able to detect previously undetected fish; however, it also introduces a few false positives.

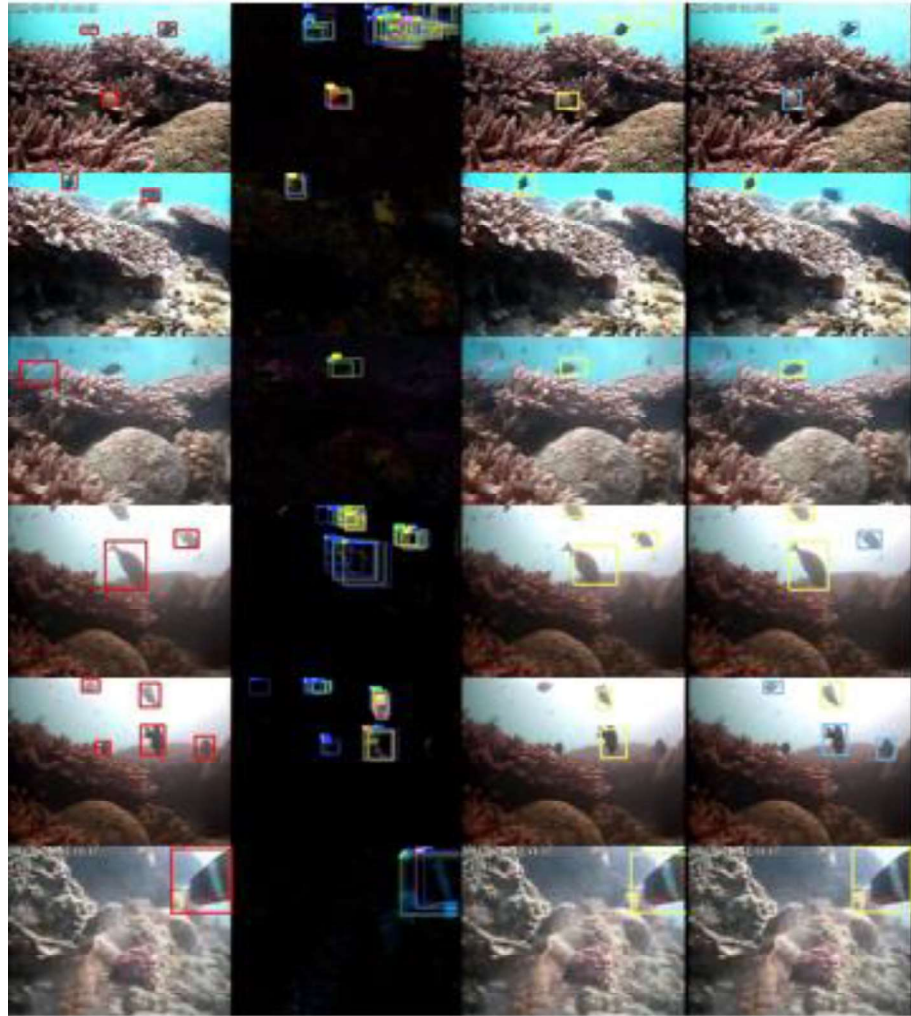


Figure 4.18: Sample cases where an auxiliary system helps with detecting previously undetected fish. From left row to right; ground truth boxes; trail image YOLO-Ang output; clustering module output; final output (the green box is from the auxiliary system; the blue box is the high-confidence detection from the original YOLOv5)

CHAPTER 5

CONCLUSION

For the task of underwater fish detection, the YOLOv5 architecture is able to detect fish; however, there is some room for improvement, especially on the number of FNs. The proposed method integrates an auxiliary system with the original YOLOv5. The auxiliary system is successful in decreasing the number of FN, albeit introducing some FP. The overall F1 score has improved by 5.28%.

This auxiliary system provides information to select low-confidence bounding boxes produced by YOLOv5, and thus it produces additional candidates (bounding boxes) for reducing FN probability. The proposed auxiliary system consists of three modules: Trail Image Formulation module, YOLO-Ang detector, and Clustering-module (with Fusion module). The Trail Image Formulation module borrows some concepts from MHI, where the frame differences of a sequence of frames are merged into a single frame, and thus the temporal information is exploited. There are a few variations of the Trail Image Formulation module: 1) Addition method, 2) Overlap method and 3) MHI. After some thorough testing, it was concluded that the addition method generally gives better performance.

The auxiliary system is trained on a dataset produced by the Trail Image Formulation module. The detector of the auxiliary system is a modification of YOLOv5 and is called YOLO-Ang. The YOLO-Ang accepts a trail image as input and produces bounding boxes of the same object in different frames. In addition, the proposed YOLO-Ang can produce angle information associated with the aforementioned bounding boxes. The outputs from YOLO-Ang often contain redundant bounding box candidates. Hence, a clustering-module is designed to merge redundant boxes and generate the supporting bounding boxes. A simple Fusion module uses the supporting boxes to pick up the low-confidence bounding boxes generated by the original YOLOv5. There are various settings with the clustering-module that are tested as well. It was found that training YOLO-Ang with angle and using confidence scores to select the final support box for the clustering-module performed the best.

In the future, code optimization for the Trail Image Formulation module, clustering-module and fusion module can be done to increase the inference speed and reduce the computational complexity.

BIBLIOGRAPHY

- [1] A. A. dos Santos and W. N. Gonçalves, 'Improving Pantanal fish species recognition through taxonomic ranks in convolutional neural networks', *Ecological Informatics*, vol. 53, p. 100977, 2019.
- [2] N. Jayachandra and A. R. Kamal, 'A novel framework for automated image set preparation for moving objects in under water videos', *International Journal of Applied Engineering Research*, vol. 12, pp. 5137–5145, 01 2017.
- [3] G. Cutter, K. Stierhoff and J. Zeng, "Automated Detection of Rockfish in Unconstrained Underwater Videos Using Haar Cascades and a New Image Dataset: Labeled Fishes in the Wild," 2015 IEEE Winter Applications and Computer Vision Workshops, Waikoloa, HI, USA, 2015, pp. 57-62, doi: 10.1109/WACVW.2015.11.
- [4] S. Zhao et al., 'Application of machine learning in intelligent fish aquaculture: A review', *Aquaculture*, vol. 540, p. 736724, 04 2021.
- [5] S. Siddiqui et al., 'Automatic fish species classification in underwater videos: Exploiting pretrained deep neural network models to compensate for limited labelled data', *ICES Journal of Marine Science*, vol. 75, 05 2017.
- [6] W. Xu and S. Matzner, "Underwater Fish Detection Using Deep Learning for Water Power Applications," MTS/IEEE OCEANS 2018 Charleston, Charleston, SC, USA, 2018, Dec. 01, 2018.
- [7] V. Shevchenko, T. Eerola and A. Kaarna, "Fish Detection from Low Visibility Underwater Videos," 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 2018, pp. 1971-1976, doi: 10.1109/ICPR.2018.8546183.
- [8] A. B. Tamou, A. Benzinou, K. Nasreddine, and L. Ballihi, 'Underwater Live Fish Recognition by Deep Learning', in *Image and Signal Processing*, 2018, pp. 275–283

- [9] M. -C. Chuang, J. -N. Hwang and K. Williams, "A Feature Learning and Object Recognition Framework for Underwater Fish Images," in *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1862-1872, April 2016, doi: 10.1109/TIP.2016.2535342.
- [10] A. Joly et al., 'LifeCLEF 2015: Multimedia Life Species Identification Challenges', in *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, 2015, pp. 462–483.
- [11] R. B. Fisher, K.-T. Shao, and Y.-H. Chen-Burger, 'Overview of the Fish4Knowledge Project', in *Fish4Knowledge: Collecting and Analyzing Massive Coral Reef Fish Video Data*, R. B. Fisher, Y.-H. Chen-Burger, D. Giordano, L. Hardman, and F.-P. Lin, Eds. Cham: Springer International Publishing, 2016, pp. 1–17.
- [12] G. Jocher, "ultralytics/yolov5," GitHub, Aug. 21, 2020.
<https://github.com/ultralytics/yolov5>
- [13] C. Cortes and V. N. Vapnik, 'Support-Vector Networks', *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [14] D. H. Ballard, 'Generalizing the Hough transform to detect arbitrary shapes', *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.
- [15] C. G. Harris and M. J. Stephens, "A Combined Corner and Edge Detector," in *Proceedings of the Alvey Vision Conference 1988*, Manchester, UK, 1988, doi: <https://doi.org/10.5244/c.2.23>.
- [16] D. Lowe, 'Distinctive Image Features from Scale-Invariant Keypoints', *International Journal of Computer Vision*, vol. 60, pp. 91–110, 11 2004.
- [17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

- [18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 'ImageNet Classification with Deep Convolutional Neural Networks', Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.
- [20] P. Soviany and R. T. Ionescu, "Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction," 2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2018, pp. 209-214, doi: 10.1109/SYNASC.2018.00041.
- [21] 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, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.
- [22] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.
- [23] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi:
- [24] W. Liu et al., 'SSD: Single Shot MultiBox Detector', in Computer Vision -- ECCV 2016, Amsterdam, The Netherlands, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2

- [25] 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), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
- [26] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.
- [27] J. Redmon and A. Farhadi, 'YOLOv3: An Incremental Improvement', arXiv e-prints, p. arXiv:1804.02767, Apr. 2018.
- [28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, 'YOLOv4: Optimal Speed and Accuracy of Object Detection', ArXiv, vol. abs/2004.10934, 2020.
- [29] T. Liang et al., "CBNet: A Composite Backbone Network Architecture for Object Detection," in IEEE Transactions on Image Processing, vol. 31, pp. 6893-6906, 2022, doi: 10.1109/TIP.2022.3216771.
- [30] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy and D. Lin, "CARAFE: Content-Aware ReAssembly of FEatures," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 3007-3016, doi: 10.1109/ICCV.2019.00310.
- [31] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, "Path Aggregation Network for Instance Segmentation," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 8759-8768, doi: 10.1109/CVPR.2018.00913.
- [32] M. Sung, S. -C. Yu and Y. Girdhar, "Vision based real-time fish detection using convolutional neural network," OCEANS 2017 - Aberdeen, Aberdeen, UK, 2017, pp. 1-6, doi: 10.1109/OCEANSE.2017.8084889.
- [33] S. Liu et al., "Embedded Online Fish Detection and Tracking System via YOLOv3 and Parallel Correlation Filter," OCEANS 2018 MTS/IEEE

Charleston, Charleston, SC, USA, 2018, pp. 1-6, doi:
10.1109/OCEANS.2018.8604658.

[34] C. Li et al., ‘YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications’, ArXiv, vol. abs/2209.02976, 2022.

[35] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, ‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors’, arXiv [cs.CV]. 2022.

[36] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics,” GitHub, Jan. 01, 2023. <https://github.com/ultralytics/ultralytics>

[37] J. Terven and D. Cordova-Esparza, ‘A Comprehensive Review of YOLO: From YOLOv1 and Beyond’, arXiv [cs.CV]. 2023.

[38] S. Goled, “How does YOLOv6 compare against YOLOv5?,” Analytics India Magazine, Jul. 01, 2022. <https://analyticsindiamag.com/how-does-yolov6-compare-against-yolov5/> (accessed Apr. 18, 2023).

[39] T. Davies, “MT-YOLOv6: A YOLO-Inspired Object Detection Model Released,” W&B, Jun. 27, 2022. <https://wandb.ai/telidavies/ml-news/reports/MT-YOLOv6-A-YOLO-Inspired-Object-Detection-Model-Released--VmlldzoyMjMzMzI5> (accessed Apr. 18, 2023).

[40] S. Rath and V. Gupta, “YOLOv5 vs YOLOv6 vs YOLOv7: Comparison of YOLO Models on Speed and Accuracy | CPU & GPU,” learnopencv.com, Nov. 29, 2022. <https://learnopencv.com/performance-comparison-of-yolo-models/#Performance-Comparison-of-YOLO-Models-for-mAP-vs-FPS> (accessed Apr. 18, 2023).

[41] S. Rath, “YOLOv6 Object Detection – Paper Explanation and Inference,” learnopencv.com, Oct. 11, 2022. <https://learnopencv.com/yolov6-object-detection/#What%E2%80%99s-New-in-YOLOv6?> (accessed Apr. 18, 2023).

- [42] A. A. Muksit, F. Hasan, M. F. Hasan Bhuiyan Emon, M. R. Haque, A. R. Anwary, and S. Shatabda, 'YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment', *Ecological Informatics*, vol. 72, p. 101847, 2022.
- [43] J. Jäger, E. Rodner, J. Denzler, V. Wolff, and K. Fricke-Neuderth, 'SeaCLEF 2016: Object Proposal Classification for Fish Detection in Underwater Videos', in *Conference and Labs of the Evaluation Forum, Evora, Portugal*, 2016.
- [44] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, 'The Pascal Visual Object Classes (VOC) Challenge', *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
- [45] Y. Wang, Q. Wang, S. Jin, W. Long, and L. Hu, 'A Literature Review of Underwater Image Detection', 02 2022.
- [46] Z. Zhao, Y. Liu, X. Sun, J. Liu, X. Yang and C. Zhou, "Composited FishNet: Fish Detection and Species Recognition From Low-Quality Underwater Videos," in *IEEE Transactions on Image Processing*, vol. 30, pp. 4719-4734, 2021, doi: 10.1109/TIP.2021.3074738.
- [47] T. Tahnim, M. M. Uddin Munna and S. M. M. Ahsan, "HOG and Color Texture Saliency: An Expedient Descriptor for Bangladeshi Fish Recognition," 2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), Gazipur, Bangladesh, 2022, pp. 1-4, doi: 10.1109/ICAEEE54957.2022.9836353.
- [48] J. Le and L. Xu, 'An Automated Fish Counting Algorithm in Aquaculture Based on Image Processing', in *Proceedings of the 2016 International Forum on Mechanical, Control and Automation (IFMCA 2016)*, 2017, pp. 358–366..
- [49] L. Yan and X. Xiang, "Application of k-NN based on Kernel in underwater target recognition," *Journal of Applied Acoustics*, vol. 38, no. 3, pp. 448–451, 2019.

- [50] Y.-H. Hsiao, C.-C. Chen, S.-I. Lin, and F.-P. Lin, 'Real-world underwater fish recognition and identification, using sparse representation', *Ecol. Informatics*, vol. 23, pp. 13–21, 2014.
- [51] S. Palazzo and F. Murabito, 'Fish Species Identification in Real-Life Underwater Images', in *Proceedings of the 3rd ACM International Workshop on Multimedia Analysis for Ecological Data*, Orlando, Florida, USA, 2014, pp. 13–18.
- [52] P. Zhuang, L. Xing, Y. Liu, S. Guo, and Y. Qiao, 'Marine Animal Detection and Recognition with Advanced Deep Learning Models', in *Conference and Labs of the Evaluation Forum*, Dublin, Ireland, 2017.
- [53] X. Li, M. Shang, J. Hao, and Z. Yang, "Accelerating fish detection and recognition by sharing CNNs with objectness learning," *OCEANS 2016 - Shanghai*, Apr. 2016, doi: <https://doi.org/10.1109/oceansap.2016.7485476>.
- [54] Y. Xue and Z. Ju, "Fish Recognition Algorithm Based on Improved AlexNet," *Electron. Sci. Technol.*, vol. 34, pp. 12–17, 2021.
- [55] N. S. Abinaya, D. Susan, and S. Rakesh Kumar, "Naive Bayesian fusion based deep learning networks for multisegmented classification of fishes in aquaculture industries," *Ecological Informatics*, vol. 61, p. 101248, Mar. 2021, doi: <https://doi.org/10.1016/j.ecoinf.2021.101248>.
- [56] C. Shi, C. Jia and Z. Chen, "FFDet: a Fully Convolutional Network for Coral Reef Fish Detection by Layer Fusion," *2018 IEEE Visual Communications and Image Processing (VCIP)*, Taichung, Taiwan, 2018, pp. 1-4, doi: [10.1109/VCIP.2018.8698738](https://doi.org/10.1109/VCIP.2018.8698738).
- [57] Z. -Y. Wu, S. -L. Tseng, H. -Y. Lin, H. -Y. Chen and T. V. Luan, "Incorporating Stereo with Convolutional Neural Networks for Real-Time Fish Detection and Classification," *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Bangkok, Thailand, 2019, pp. 83-88, doi: [10.1109/CIS-RAM47153.2019.9095805](https://doi.org/10.1109/CIS-RAM47153.2019.9095805).

- [58] S. Li, B. Pan, Y. Cheng, X. Yan, C. Wang, and C. Yang, "Underwater Fish Object Detection based on Attention Mechanism improved Ghost-YOLOv5," 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, Apr. 2022, doi: <https://doi.org/10.1109/icsp54964.2022.9778582>.
- [59] X. Hu et al., 'Real-time detection of uneaten feed pellets in underwater images for aquaculture using an improved YOLO-V4 network', *Computers and Electronics in Agriculture*, vol. 185, p. 106135, 2021.
- [60] E. Prasetyo, N. Suciati, and C. Fatchah, 'Multi-level residual network VGGNet for fish species classification', *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 8, Part A, pp. 5286–5295, 2022.
- [61] Z. Zhang, X. Du, L. Jin, S. Wang, L. Wang, and X. Liu, 'Large-scale underwater fish recognition via deep adversarial learning', *Knowledge and Information Systems*, vol. 64, pp. 353–379, 2022.
- [62] S. Villon et al., 'A Deep learning method for accurate and fast identification of coral reef fishes in underwater images', *Ecological Informatics*, vol. 48, pp. 238–244, 2018.
- [63] A. Jalal, A. Salman, A. Mian, M. Shortis, and F. Shafait, 'Fish detection and species classification in underwater environments using deep learning with temporal information', *Ecological Informatics*, vol. 57, p. 101088, 2020.
- [64] A. Ben Tamou, A. Benzinou, and K. Nasreddine, 'Multi-Stream Fish Detection in Unconstrained Underwater Videos by the Fusion of Two Convolutional Neural Network Detectors', *Applied Intelligence*, vol. 51, no. 8, pp. 5809–5821, Aug. 2021.
- [65] A. Salman et al., 'Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system', *ICES Journal of Marine Science: journal du conseil*, vol. 77, no. 4, pp. 1295–1307, Jul. 2020.

- [66] Z. Shen and C. Nguyen, "Temporal 3D RetinaNet for fish detection," 2020 Digital Image Computing: Techniques and Applications (DICTA), Melbourne, Australia, 2020, pp. 1-5, doi: 10.1109/DICTA51227.2020.9363372.
- [67] A. B. Labao and P. C. Naval, 'Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild', *Ecological Informatics*, vol. 52, pp. 103–121, 2019.
- [68] T.-Y. Lin et al., 'Microsoft COCO: Common Objects in Context', in *Computer Vision -- ECCV 2014*, 2014, pp. 740–755.
- [69] M. A. R. Ahad, 'Motion History Images for Action Recognition and Understanding', in *SpringerBriefs in Computer Science*, 2012.
- [70] J. W. Davis, A. M. Morison and D. D. Woods, "Building Adaptive Camera Models for Video Surveillance," 2007 IEEE Workshop on Applications of Computer Vision (WACV '07), Austin, TX, USA, 2007, pp. 34-34, doi: 10.1109/WACV.2007.16.
- [71] M. A. R. Ahad, J. K. Tan, H. Kim, and S. Ishikawa, 'Motion History Image: Its Variants and Applications', *Mach. Vision Appl.*, vol. 23, no. 2, pp. 255–281, Mar. 2012
- [72] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257-267, March 2001, doi: 10.1109/34.910878.
- [73] J. W. Davis, "Appearance-based motion recognition of human actions," Doctoral dissertation, Massachusetts Institute of Technology, 1996.
- [74] J. R. Bergen, P. J. Burt, R. Hingorani and S. Peleg, "A three-frame algorithm for estimating two-component image motion," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 886-896, Sept. 1992, doi: 10.1109/34.161348.

- [75] M. A. R. Ahad, J. K. Tan, H. S. Kim, and S. Ishikawa, 'Temporal Motion Recognition and Segmentation Approach', *Int. J. Imaging Syst. Technol.*, vol. 19, no. 2, pp. 91–99, Jun. 2009.
- [76] ACCV'07 Workshop on Multi-dimensional and Multi-view Image Processing
- [77] M. Valstar, M. Pantic and I. Patras, "Motion history for facial action detection in video," 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, Netherlands, 2004, pp. 635-640 vol.1, doi: 10.1109/ICSMC.2004.1398371.
- [78] H. Meng, N. Pears and C. Bailey, "A Human Action Recognition System for Embedded Computer Vision Application," 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 2007, pp. 1-6, doi: 10.1109/CVPR.2007.383420.
- [79] P. S. Chandragiri and E. Paul Ijjina, "Recognizing Human Actions in Video using Motion History Image and Deep Learning," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 01-05, doi: 10.1109/ICCCNT51525.2021.9579817.
- [80] O. Mercanoglu Sincan and H. Y. Keles, "Using Motion History Images With 3D Convolutional Networks in Isolated Sign Language Recognition," in *IEEE Access*, vol. 10, pp. 18608-18618, 2022, doi: 10.1109/ACCESS.2022.3151362.
- [81] I. T. Toudjeu and J. -R. Tapamo, "A 2D Convolutional Neural Network Approach for Human Action Recognition," 2019 IEEE AFRICON, Accra, Ghana, 2019, pp. 1-5, doi: 10.1109/AFRICON46755.2019.9133840.
- [82] T. Kurita, N. Otsu, and N. Abdelmalek, 'Maximum likelihood thresholding based on population mixture models', *Pattern Recognition*, vol. 25, no. 10, pp. 1231–1240, 1992

- [83] T. F. Dima and M. E. Ahmed, "Using YOLOv5 Algorithm to Detect and Recognize American Sign Language," 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 2021, pp. 603-607, doi: 10.1109/ICIT52682.2021.9491672.
- [84] M. Feurer and F. Hutter, 'Hyperparameter Optimization', in Automated Machine Learning: Methods, Systems, Challenges, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.
- [85] F. Hutter, H. H. Hoos, and K. Leyton-Brown, 'Sequential Model-Based Optimization for General Algorithm Configuration', in Proceedings of the 5th International Conference on Learning and Intelligent Optimization, Rome, Italy, 2011, pp. 507–523.
- [86] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, 'Algorithms for Hyper-Parameter Optimization', in Proceedings of the 24th International Conference on Neural Information Processing Systems, Granada, Spain, 2011, pp. 2546–2554.
- [87] J. Snoek, H. Larochelle, and R. P. Adams, 'Practical Bayesian Optimization of Machine Learning Algorithms', in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, Lake Tahoe, Nevada, 2012, pp. 2951–2959.
- [88] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, 'Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms', in Proc. of KDD-2013, 2013, pp. 847–855
- [89] J. Bergstra, D. Yamins, and D. D. Cox, 'Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures', in Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, 2013, Atlanta, GA, USA, p. I-115-I-123
- [90] "Thop," [Library description] PyPI, <https://pypi.org/project/thop/> (accessed Jul. 18, 2023).

- [91] Ultralytics, “Release v3.0 · ultralytics/yolov5,” GitHub, <https://github.com/ultralytics/yolov5/releases/tag/v3.0> (accessed Jul. 18, 2023).
- [92] Ultralytics, “Release v4.0 - nn.silu() activations, weights & biases logging, pytorch hub integration · ultralytics/yolov5,” GitHub, <https://github.com/ultralytics/yolov5/releases/tag/v4.0> (accessed Jul. 18, 2023).
- [93] Ultralytics, “Ultralytics/Yolov5 at 73a066993051339f6adfe5095a7852a2b9184c16,” GitHub, <https://github.com/ultralytics/yolov5/tree/73a066993051339f6adfe5095a7852a2b9184c16> (accessed Jul. 18, 2023).

APPENDIX

Tables A2 to A5 show the results for the system using trail formulation with addition method and overlap method, respectively for 15 species. The “Total TP” and “Total FP” columns in the tables simply represent the total number of TP and FP output produced by the whole system. The calculation includes adding extra TP and extra FP (from the auxiliary system) with the results from the original YOLOv5 RGB output. The original TP, FP and FN is 6574, 4014, and 5115, respectively and the F1 score is 59.02. All of the results shown in the tables below use the default parameters.

Table A1: Description and respective table number for referencing convenience

Trail formulation method	Model type	Table number
Addition	$nf(n-1)bb$	A2
	$nf nbb$	A3
Overlap	$nf(n-1)bb$	A4
	$nf nbb$	A5

Table A2: Variations with angle information using $nf(n-1)bb$ trail image with addition method.

$Nf(n-1)bb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (ExtraTP + 6574)	Total FP (Extra FP+ 4013)	Total FN (5115- Extra TP)	Total no. of extra boxes	Extra TP out of extras %
3f2bb	With angle	With angle	61.85 (+2.83)	1003	1222	7577	5235	4112	2225	45.08
4f3bb			62.55 (+3.53)	1183	1342	7757	5355	3932	2525	46.85
5f4bb			63.38 (+4.36)	1320	1313	7894	5326	3795	2633	50.13
6f5bb			63.1 (+4.08)	1640	2117	8214	6130	3475	3757	43.65
7f6bb			63.56 (+4.54)	1457	1537	8031	5550	3658	2994	48.66
8f7bb			63.67 (+4.65)	1884	2410	8458	6423	3231	4294	43.88
9f8bb			63.15 (+4.13)	1606	2025	8180	6038	3509	3631	44.23
3f2bb	With angle	No angle	62.06 (+3.04)	1100	1353	7674	5366	4015	2453	44.84
4f3bb			62.88 (+3.86)	1286	1439	7860	5452	3829	2725	47.19
5f4bb			63.46 (+4.44)	1361	1369	7935	5382	3754	2730	49.85
6f5bb			63.76 (+4.74)	1849	2295	8423	6308	3266	4144	44.62
7f6bb			63.95 (+4.93)	1582	1649	8156	5662	3533	3231	48.96
8f7bb			64.24 (+5.22)	2097	2624	8671	6637	3018	4721	44.42
9f8bb			63.72 (+4.7)	1788	2183	8362	6196	3327	3971	45.03
3f2bb	No angle	No angle	60.53 (+1.51)	410	389	6984	4402	4705	799	51.31
4f3bb			60.28 (+1.26)	345	336	6919	4349	4770	681	50.66
5f4bb			63.24 (+4.22)	1286	1297	7860	5310	3829	2583	49.79
6f5bb			63.84 (+4.82)	1292	1075	7866	5088	3823	2367	54.58
7f6bb			63.94 (+4.92)	1484	1445	8058	5458	3631	2929	50.67
8f7bb			63.45 (+4.43)	1411	1483	7985	5496	3704	2894	48.76
9f8bb			62.95 (+3.93)	1227	1281	7801	5294	3888	2508	48.92

Table A3: Variations with angle information using $nfnbb$ trail image with addition method.

$Nfnbb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (ExtraTP + 6574)	Total FP (Extra FP+ 4013)	Total FN (5115- Extra TP)	Total no. of extra boxes	Extra TP out of extras %
3f3bb	With angle	With angle	62.03 (+3.01)	1022	1193	7596	5206	4093	2215	46.14
4f4bb			63.45 (+4.43)	1389	1435	7963	5448	3726	2824	49.19
5f5bb			63.97 (+4.95)	1485	1435	8059	5448	3630	2920	50.86
6f6bb			63.81 (+4.79)	1658	1868	8232	5881	3457	3526	47.02
7f7bb			63.62 (+4.6)	1360	1304	7934	5317	3755	2664	51.05
8f8bb			63.95 (+4.93)	1605	1698	8179	5711	3510	3303	48.59
9f9bb			63.03 (+4.01)	1285	1375	7859	5388	3830	2660	48.31
3f3bb	With angle	No angle	62.34 (+3.32)	1117	1281	7691	5294	3998	2398	46.58
4f4bb			63.72 (+4.7)	1459	1479	8033	5492	3656	2938	49.66
5f5bb			64.18 (+5.16)	1540	1470	8114	5483	3575	3010	51.16
6f6bb			64.3 (+5.28)	1831	2036	8405	6049	3284	3867	47.35
7f7bb			63.94 (+4.92)	1445	1361	8019	5374	3670	2806	51.5
8f8bb			64.3 (+5.28)	1716	1795	8290	5808	3399	3511	48.87
9f9bb			63.3 (+4.28)	1360	1431	7934	5444	3755	2791	48.73
3f3bb	No angle	No angle	59.92 (+0.9)	236	220	6810	4233	4879	456	51.75
4f4bb			62.06 (+3.04)	770	620	7344	4633	4345	1390	55.4
5f5bb			63.11 (+4.09)	1137	1022	7711	5035	3978	2159	52.66
6f6bb			64.14 (+5.12)	1745	1919	8319	5932	3370	3664	47.63
7f7bb			63.24 (+4.22)	1130	958	7704	4971	3985	2088	54.12
8f8bb			63.84 (+4.82)	1474	1462	8048	5475	3641	2936	50.2
9f9bb			63.1 (+4.08)	1298	1377	7872	5390	3817	2675	48.52

Table A4: Variations with angle information using $n(n-1)bb$ trail image with overlap method.

$N(n-1)bb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (Extra TP + 6574)	Total FP (Extra FP + 4013)	Total FN (5115 - Extra TP)	Total no. of extra boxes	Extra TP out of extras %		
3f2bb	With angle	With angle	62.72 (+3.7)	1664	2330	8238	6343	3451	3994	41.66		
4f3bb			62.92 (+3.9)	1526	1943	8100	5956	3589	3469	43.99		
5f4bb			62.88 (+3.86)	1519	1948	8093	5961	3596	3467	43.81		
6f5bb			62.9 (+3.88)	1352	1572	7926	5585	3763	2924	46.24		
7f6bb			63.1 (+4.08)	1271	1318	7845	5331	3844	2589	49.09		
8f7bb			63.19 (+4.17)	1326	1400	7900	5413	3789	2726	48.64		
9f8bb			63.21 (+4.19)	1349	1442	7923	5455	3766	2791	48.33		
3f2bb			With angle	No angle	63.48 (+4.46)	1917	2560	8491	6573	3198	4477	42.82
4f3bb					63.48 (+4.46)	1691	2073	8265	6086	3424	3764	44.93
5f4bb	63.52 (+4.5)	1673			2018	8247	6031	3442	3691	45.33		
6f5bb	63.3 (+4.28)	1460			1647	8034	5660	3655	3107	46.99		
7f6bb	63.29 (+4.27)	1340			1391	7914	5404	3775	2731	49.07		
8f7bb	63.43 (+4.41)	1401			1470	7975	5483	3714	2871	48.8		
9f8bb	63.17 (+4.15)	1375			1516	7949	5529	3740	2891	47.56		
3f2bb	No angle	No angle			62.39 (+3.37)	1264	1585	7838	5598	3851	2849	44.37
4f3bb					62.14 (+3.12)	935	956	7509	4969	4180	1891	49.44
5f4bb			63.21 (+4.19)	1570	1923	8144	5936	3545	3493	44.95		
6f5bb			63.32 (+4.3)	1390	1489	7964	5502	3725	2879	48.28		
7f6bb			63.68 (+4.66)	1613	1822	8187	5835	3502	3435	46.96		
8f7bb			63.36 (+4.34)	1291	1260	7865	5273	3824	2551	50.61		
9f8bb			62.13 (+3.11)	933	955	7507	4968	4182	1888	49.42		

Table A5: Variations with angle information using $nfnbb$ trail image with overlap method.

$Nfnbb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (Extra TP + 6574)	Total FP (Extra FP + 4013)	Total FN (5115 - Extra TP)	Total no. of extra boxes	Extra TP out of extras %		
3f3bb	With angle	With angle	62.76 (+3.74)	1575	2117	8149	6130	3540	3692	42.66		
4f4bb			62.54 (+3.52)	1367	1752	7941	5765	3748	3119	43.83		
5f5bb			63.66 (+4.64)	1642	1894	8216	5907	3473	3536	46.44		
6f6bb			63.33 (+4.31)	1515	1754	8089	5767	3600	3269	46.34		
7f7bb			63.15 (+4.13)	1382	1541	7956	5554	3733	2923	47.28		
8f8bb			63.04 (+4.02)	1362	1538	7936	5551	3753	2900	46.97		
9f9bb			62.99 (+3.97)	1290	1402	7864	5415	3825	2692	47.92		
3f3bb			With angle	No angle	63.46 (+4.44)	1788	2291	8362	6304	3327	4079	43.83
4f4bb					63.05 (+4.03)	1519	1877	8093	5890	3596	3396	44.73
5f5bb	64.09 (+5.07)	1774			1999	8348	6012	3341	3773	47.02		
6f6bb	63.71 (+4.69)	1610			1807	8184	5820	3505	3417	47.12		
7f7bb	63.46 (+4.44)	1468			1603	8042	5616	3647	3071	47.8		
8f8bb	63.36 (+4.34)	1441			1583	8015	5596	3674	3024	47.65		
9f9bb	63.23 (+4.21)	1355			1450	7929	5463	3760	2805	48.31		
3f3bb	No angle	No angle			63.13 (+4.11)	1623	2069	8197	6082	3492	3692	43.96
4f4bb					63.23 (+4.21)	1403	1552	7977	5565	3712	2955	47.48
5f5bb			62.67 (+3.65)	1076	1062	7650	5075	4039	2138	50.33		
6f6bb			62.61 (+3.59)	1099	1134	7673	5147	4016	2233	49.22		
7f7bb			62.58 (+3.56)	1055	1051	7629	5064	4060	2106	50.09		
8f8bb			62.81 (+3.79)	1217	1315	7791	5328	3898	2532	48.06		
9f9bb			62.18 (+3.16)	1009	1106	7583	5119	4106	2115	47.71		

Tables A6 and 4.A7 show the results with trail formulation using the MHI method. The post-processing module parameters use the default values.

Table A6: Variations with angle information using $nf(n-1)bb$ MHI.

$Nf(n-1)bb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (Extra TP + 6574)	Total FP (Extra FP + 4013)	Total FN (5115 - Extra TP)	Total no. of extra boxes	Extra TP out of extras %
3f2bb	With angle	With angle	61.16 (+2.14)	1669	3010	8243	7023	3446	4679	35.67
4f3bb			61.8 (+2.78)	1186	1653	7760	5666	3929	2839	41.78
5f4bb			61.96 (+2.94)	1591	2490	8165	6503	3524	4081	38.99
6f5bb			62.38 (+3.36)	1447	1994	8021	6007	3668	3441	42.05
3f2bb	With angle	No angle	61.67 (+2.65)	1959	3437	8533	7450	3156	5396	36.3
4f3bb			62.22 (+3.2)	1318	1776	7892	5789	3797	3094	42.6
5f4bb			62.47 (+3.45)	1774	2675	8348	6688	3341	4449	39.87
6f5bb			62.6 (+3.58)	1546	2122	8120	6135	3569	3668	42.15
3f2bb	No angle	No angle	62.42 (+3.4)	1473	2036	8047	6049	3642	3509	41.98
4f3bb			62.8 (+3.78)	1703	2380	8277	6393	3412	4083	41.71
5f4bb			61.59 (+2.57)	2057	3696	8631	7709	3058	5753	35.76
6f5bb			61.84 (+2.82)	1036	1300	7610	5313	4079	2336	44.35

Table A7: Variations with angle information using $nfnbb$ MHI.

$Nfnbb$	Model angle status	Clustering module	F1 score (improvement)	Extra TP	Extra FP	Total TP (Extra TP + 6574)	Total FP (Extra FP + 4013)	Total FN (5115 - Extra TP)	Total no. of extra boxes	Extra TP out of extras %
3f3bb	With angle	With angle	62.23 (+3.21)	1303	1736	7877	5749	3812	3039	42.88
4f4bb			61.54 (+2.52)	1135	1643	7709	5656	3980	2778	40.86
5f5bb			62.03 (+3.01)	1602	2482	8176	6495	3513	4084	39.23
6f6bb			61.94 (+2.92)	1178	1578	7752	5591	3937	2756	42.74
3f3bb	With angle	No angle	62.67 (+3.65)	1435	1849	8009	5862	3680	3284	43.7
4f4bb			61.89 (+2.87)	1231	1716	7805	5729	3884	2947	41.77
5f5bb			62.61 (+3.59)	1711	2478	8285	6491	3404	4189	40.85
6f6bb			62.25 (+3.23)	1258	1629	7832	5642	3857	2887	43.57
3f3bb	No angle	No angle	62.07 (+3.05)	1198	1568	7772	5581	3917	2766	43.31
4f4bb			62.23 (+3.21)	1313	1760	7887	5773	3802	3073	42.73
5f5bb			62.01 (+2.99)	1127	1435	7701	5448	3988	2562	43.99
6f6bb			61.8 (+2.78)	907	1028	7481	5041	4208	1935	46.87