TKINTER-BASED DESKTOP APPLICATION FOR FISH RECOGNITION USING HAAR CASCADE DETECTION

A PROJECT REPORT

Submitted by

YESHWANTH J(RA2111003040076)

Under the guidance of

Dr.N.MUTHURASU

(Assistant Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956
VADAPALANI

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

VADAPALANI – 600 026

May 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report titled "**TKINTER-BASED DESKTOP APPLICATION FOR FISH RECOGNITION USING HAAR CASCADE DETECTION**" is the bonafide work of "**YESHWANTH J[RA2111003040076]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE
Dr.N.MUTHURASU
Assistant Professor
Dept. of Computer Science & Engg,
SRM IST
Vadapalani Campus

HEAD OF THE DEPARTMENT
Dr. GOLDA DILIP
Professor and Head,
Dept. of Computer Science & Engg,
SRM IST,
Vadapalani Campus

Signature of the Internal Examiner

Signature of the External Examiner

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done

To be completed by the student for all assessments

Student Name : YESHWANTH J

Reg. Number : RA2111003040076

Title of Work : TKINTER-BASED DESKTOP APPLICATION FOR FISH RECOGNITION USING HAAR CASCADE DETECTION

Degree/Course : B.TECH/CSE

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website.
- I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except were indicated by referring, and that I have followed the good academic practices noted above

J Yeshwanth	G Wilfred Auxilian	C Abishek
(RA2111003040076)	(RA2111003040077)	(RA21110030400109)

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENTS

We express our humble gratitude to our Honorable Chancellor **Dr. T. R. Paarivendhar**, Pro Chancellor (Administration), **Dr. Ravi Pachamoothu**, Pro Chancellor (Academics) **Dr.P. Sathyanarayanan**, Pro Chancellor (Admin) for the facilities extended for the completion of the project work.

We would record our sincere gratitude to our Vice Chancellor, **Dr.C. Muthamizhchelvan** and Registrar, **Dr. S. Ponnusamy** for their support to complete our project work by giving us the best of academic excellence support system in place. We extend our sincere thanks to our Dean, **Dr.C V Jayakumar** and Vice Principal – Academics, **Dr.C. Gomathy** and Vice Principal Examination **Dr.S. Karthikeyan** for their invaluable support.

We wish to thank **Dr.Golda Dilip** , Professor & Head, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for her valuable suggestions and encouragement throughout the period of the project work and the course.

We are extremely grateful to our Project Coordinator, **Dr.P. Durgadevi**, Assistant Professor, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus, for leading and helping us to complete our course.

We extend our gratitude to our supervisor, **Dr.N.Muthurasu**, Assistant Professor, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for providing us an opportunity to pursue our project under her mentorship. She provided us the freedom and support to explore the research topics of our interest.

We sincerely thank all faculty of DCSE, staff and students of the department who have directly or indirectly helped our project.

Finally, we would like to thank our parents, our family members and our friends for their unconditional love, constant support, and encouragement.

YESHWANTH J
(RA2111003040076)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This research introduces a hybrid framework for underwater fish detection which combines three powerful object detection algorithms for improving the accuracy of detection speed for many different species of fish underwater. Each of these algorithms are different from each other with respect to accuracy detection capabilities and speed of detecting an object. The inclusion of these algorithms in a hybrid framework is necessary to overcome the challenges of undertaking fish detection in underwater environments, specifically low dim lighting situations, color distortion, and high turbidity, which adds distortion to fish identification frequency. The image enhancement pipeline includes enough methods to apply exposures to diminish the impacts of the three effectors scalably: white balancing, histogram equalization, and CLAHE and Each of these methods work simultaneously in elongated signal processing to constitute more clarification of the image and improve the contrast of the image from the initial image, increasing accuracy of detection identification to detect the fish in the underwater environmental constraint challenges.

The hybrid framework's efficacy is systematically evaluated and experiments comparing the hybrid framework and traditional single model methods are documented. The hybrid framework demonstrates substantial increases in accuracy and processing speed, and highlights the merits of a hybrid approach to embrace complex scenarios in the underwater domain.Given that we implemented advanced artificial intelligence (AI) approaches, this project offered a new way to detect fish with greater detection efficacy, while providing more efficient management in aquatic systems. Future development will involve developing new algorithms and expanding the dataset, which will lead to improved performance with detection and adaptation to varied underwater conditions.

CHAPTER 1

1. INTRODUCTION

1.1. Overview of Underwater Fish Detection: -

- Underwater fish detection is an important aspect of fish study and biodiversity documentation.

- Conventional methods of fish detection such as manual underwater surveys and netting are laborious and can disturb the marine environment.

- The demand for automated detection of fish populations is increasing to detect and monitor fish populations with efficiency and accuracy in difficult underwater environments.

  Underwater fish detection is a critical component in marine science for gathering important information about aquatic ecosystems. With a growing demand for sustainable, viable fisheries and conservation, the ability to accurately monitor fish populations has great significance.

1.2. Importance of AI in Marine Biology: -

- Deep learning and computer vision are AI-based approaches that can increase the accuracy and consistency of fish identification, allowing researchers to process thousands of minutes of underwater footage and images to piece together species identification and track fish populations over time.

- AI has fundamentally changed the process marine biologist are able to analyze and understand data, as it added additional level of capacity and pattern recognition in the data.

- AI will allow researchers to analyze a huge quantity of visual data which will potentially provide greater precision of information linked to fish populations and habitat conditions.

1.3. Objectives of the Study: -

- To propose a hybrid framework for underwater fish detection, fused with YOLOv11, Faster R-CNN, and Haar Cascade.

- To improve detection accuracy and processing speed under complex underwater scenarios/conditions.

- To deploy an image enhancement pipeline as an option to enhance the quality of underwater images.

CHAPTER 2

2. LITERATURE SURVEY

2.1. Previous Research on Fish Detection Techniques: -

Author	Method / Model	Technique / Approach	Key Insights
Kazim Raza , Song Hong (2020) [1]	Improved YOLOv3	Transfer learning for fast detection	Enhanced speed and accuracy in fish detection
Chunqi Gao et al. (2022) [2]	YOLOv3, YOLOv4	Accurate detection with attention mechanism	Improved individual fish recognition
Ahsan Jalal et al. (2025) [3]	DeepFins	Dynamic video analysis for fish detection	Captures underwater video dynamics
K. Prabhakaran et al. L. (2022) [4]	Haar-Cascade	Template matching for poly-metallic nodules	Effective detection of nodules in underwater images
Musab Iqtait et al. (2024) [5]	Enhanced Deep Learning	Novel approach with multi-scale features	Achieved high accuracy in fish species classification
R. M. Harish et al. (2022) [6]	Image Processing	Turbidity level quantification	Quantified relative turbidity using image techniques
Jenish Savaliya et al. (2023) [7]	Image Processing	Detection of underwater resources	Focused on fauna, flora, and nodules with GUI support
Pratima Sarkar et al. (2023) [8]	YOLOv3, YOLOv4	Real time detection with Darknet-53	Achieved 50% MAP (Mean Average Precision); highlighted challenges in data balance

2.2. Comparison of Existing models: -

Parameter	YOLOv11	Faster R-CNN	Haar Cascade
Use Case	Real-time object detection	Precise object detection	Simple object detection
Dataset Preparation	Requires bounding box annotations	High quality, annotated data	Pretrained Haar models or manual samples
Hardware Requirements	Moderate (GPU recommended, e.g.:16GB+ RAM)	High (powerful GPUs needed)	Low (CPU with 4GB+ RAM)
Challenges	Requires tuning for small underwater objects; higher resource demand than YOLOv5	Computationally expensive; slow for real-time	Poor performance in murky water or complex scenes
Notes	Latest YOLO version with enhanced speed and accuracy; ideal for real time underwater tasks	Great for scientific underwater monitoring and taxonomy	Limited to simple tasks like diver face detection underwater
Custom Data Acceptance	Yes	Yes	Yes (with manual preparation)
Dataset Preparation Tool	LabelImg, Roboflow, CVAT, etc.	TensorFlow Datasets, Roboflow, OpenCV	LabelImg, OpenCV_createsamples
Performance	Superior speed and accuracy; optimized for realtime with enhanced features	High accuracy but slow inference; robust for complex tasks	Lightweight but poor accuracy on diverse datasets
Desktop Integration	Yes	Yes	Yes
Libraries Required	OpenCV, PyTorch, NumPy, etc.	TensorFlow, OpenCV, NumPy	OpenCV, NumPy
Features Available	Real-time detection, segmentation, pose estimation	Scalability, region proposals	Simple feature-based detection
Concept	Single stage detection with advanced feature extraction	Two stage detection with region proposals	Cascade of classifiers on Haar features
Challenges (Detailed)	Tuning for small objects, resource-intensive training	High learning curve and resource-heavy	Labor intensive prep but outdated

2.3. Challenges in Underwater Imaging: -

- ❖ Low-light situations may lead to unacceptable visibility, and subsequently, low-level, poor-quality, images.

- ❖ Water absorption and scattering may cause color distortion, like reds and greens in deeper water.

- ❖ High turbidity associated with are often accompanied high concentrations of suspended particles which could readily distort images making fish identification difficult.

- ❖ Reflections and glare bouncing off a water surface can potentially affect the quality of all images and the success of all detection efforts.

- ❖ Air bubbles in the water may obstruct views to fish subject but may also create noise types in images potentially affecting all detection efforts.

- ❖ Varying water conditions can affect both views and fish behavior, e.g., currents, temperature gradients etc.

- ❖ Very complex backgrounds consisting of rocks, plants, refuse etc can obstruct any feature detection algorithms.

- ❖ Fish demonstrating erratic/non-linear behavior can reduce or eliminate clear images and their proper identification.

- ❖ Limited field of view ("V") may exclude some fish outside the camera's "V".

- ❖ Equipment limitations caused by water pressure, temperature, and salinity may also influence performance and image quality.

CHAPTER 3

3. METHODLOGY

3.1. Hybrid Framework Overview: -

The hybrid framework proposal integrates various object detection algorithms and image enhancement methods to improve accuracy and processing speed for underwater fish detection. This hybrid framework aims to tackle the challenges of underwater imaging by using a mix of models and to improve image quality prior to typical object detection tasks. The overall architecture will consist of a preprocessing phase for the image that involves some form of imaging enhancement step followed by the application of object detection algorithm step to detect and classify fish species in real-time.

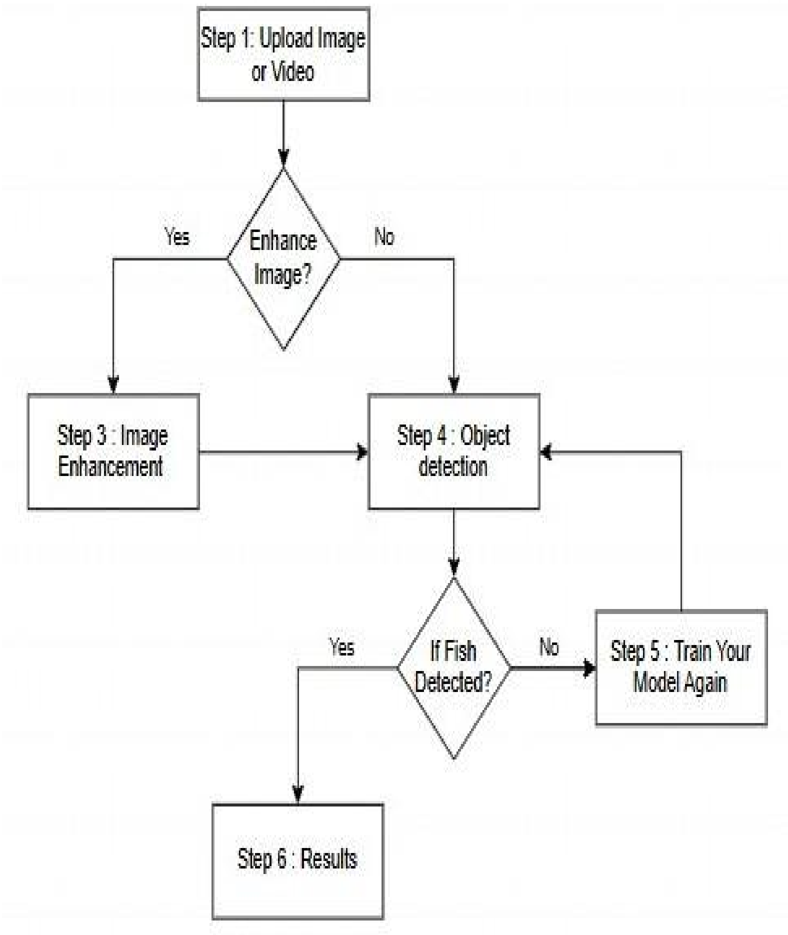


Figure 3.1: The overall basic workflow diagram

3.2. Object Detection Algorithms: -

This framework consists of three detection algorithms, each of which were selected for their unique benefits for detecting fish in underwater environments. The flowchart below presents the typical system architecture diagram for each of the 3 models.

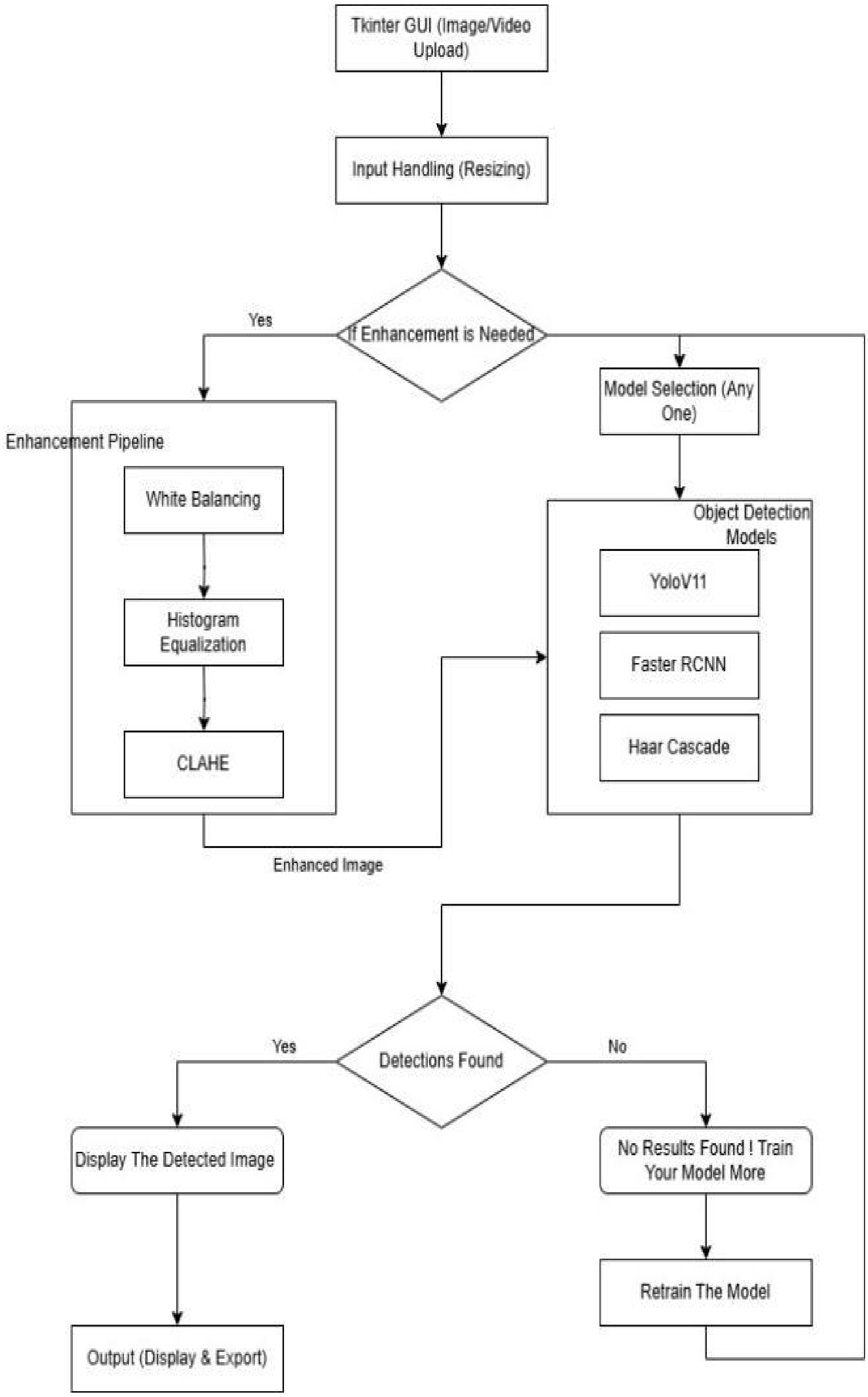


Figure 3.2: The overall advanced workflow diagram

3.2.1. Yolov11: -

YOLOv11 (You Only Look Once version 11) is a modern state of the art real-time object detection model with a value of rapid processing time. YOLOv11 can also take an image and predict bounding boxes and class probabilities in only one run of the model through a single network evaluation. These characteristics make the YOLOv11 unique and able to meet our need for detecting fish as quick as possible underwater. YOLOv11 is trained off a larger dataset of underwater images to maximize its ability to detect the various species of fish.

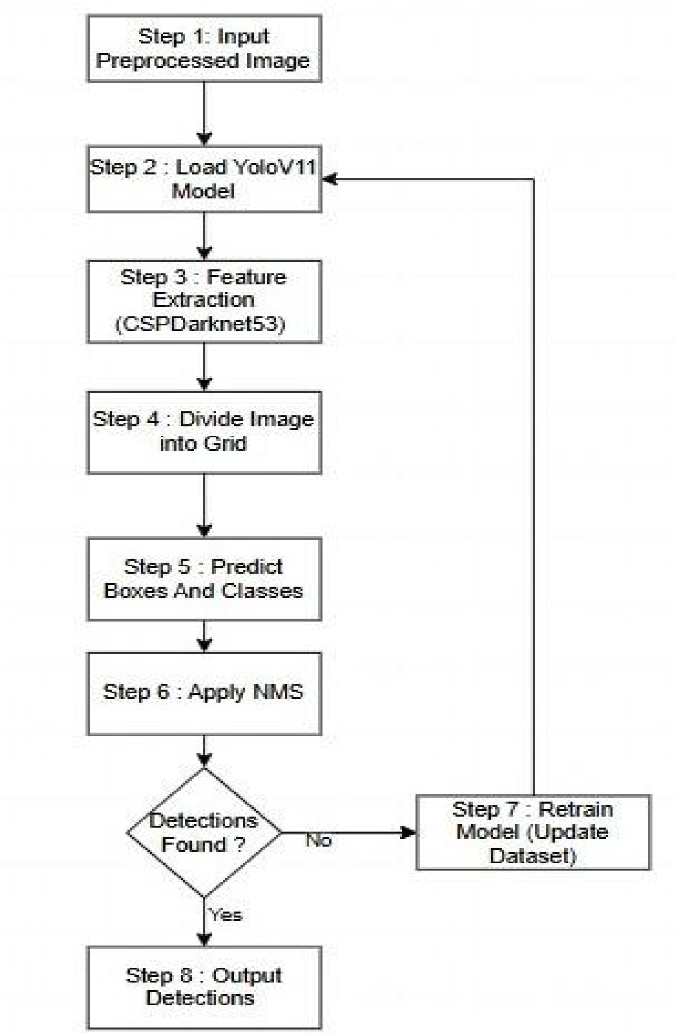


Figure 3.2.1:- The Flowchart of yolov11 detection model

3.2.2. FasterRCNN: -

Faster R-CNN (Region based Convolutional Neural Networks) is another state of the art object detection algorithm that is well known for its accuracy. It is a two-stage model where the first stage looks at proposed region locations and the second stage classifies the regions. The resource of the Architecture and Compunding Power of R-CNN will cost more than YOLO but also being more advantageous because more accurate and useful for detecting smaller fish or fish overlapping each other. The R-CNN model is updated model with undetermined annotated datasets to maximize it ability in detecting fish and recognizing in a real life scenario and then we can observe the results.

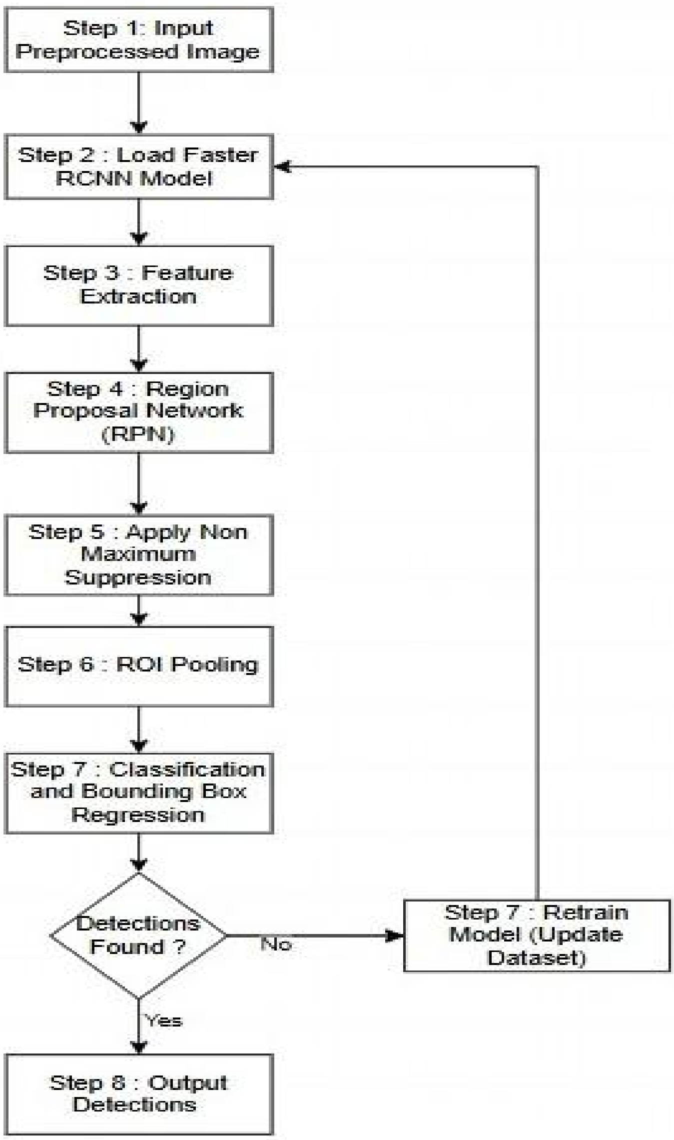


Figure 3.2.2:- Flowchart of faster CNN detection model

3.2.3. Haar Cascade: -

Haar Cascade is a basic object detection technique that uses features to detect objects. It is simple and fast and is used in applications where computational power is limited. Haar Cascade will not necessarily yield the accuracy of YOLOv11 & Faster R-CNN but can be used alongside other work, detecting special species of fish and used for a similar effort in less complicated situations. Clearly, the model must be trained with a set of positive and negative images to generate a classifier.

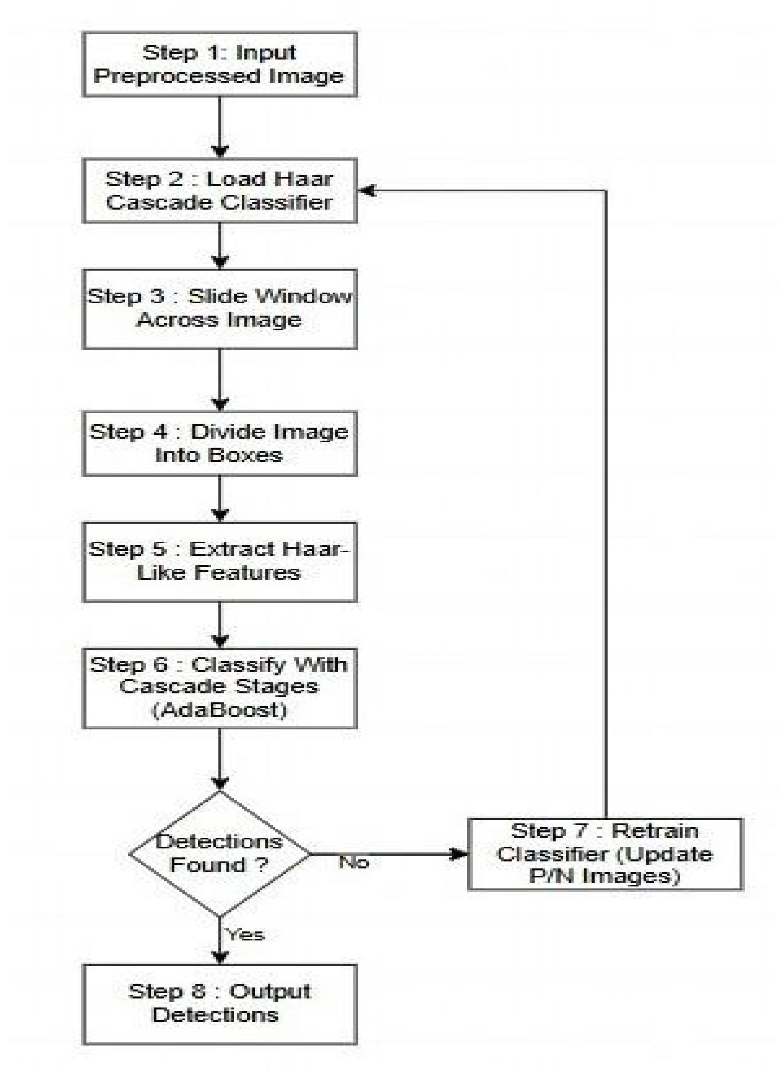


Figure 3.2.3:- Flow chart of the haar cascade detection model

3.3. Image Enhancement Techniques: -

Several image enhancement techniques are applied to the underwater images prior to implementing any object detection algorithm to optimize low-light image quality.

3.3.1. White Balance: -

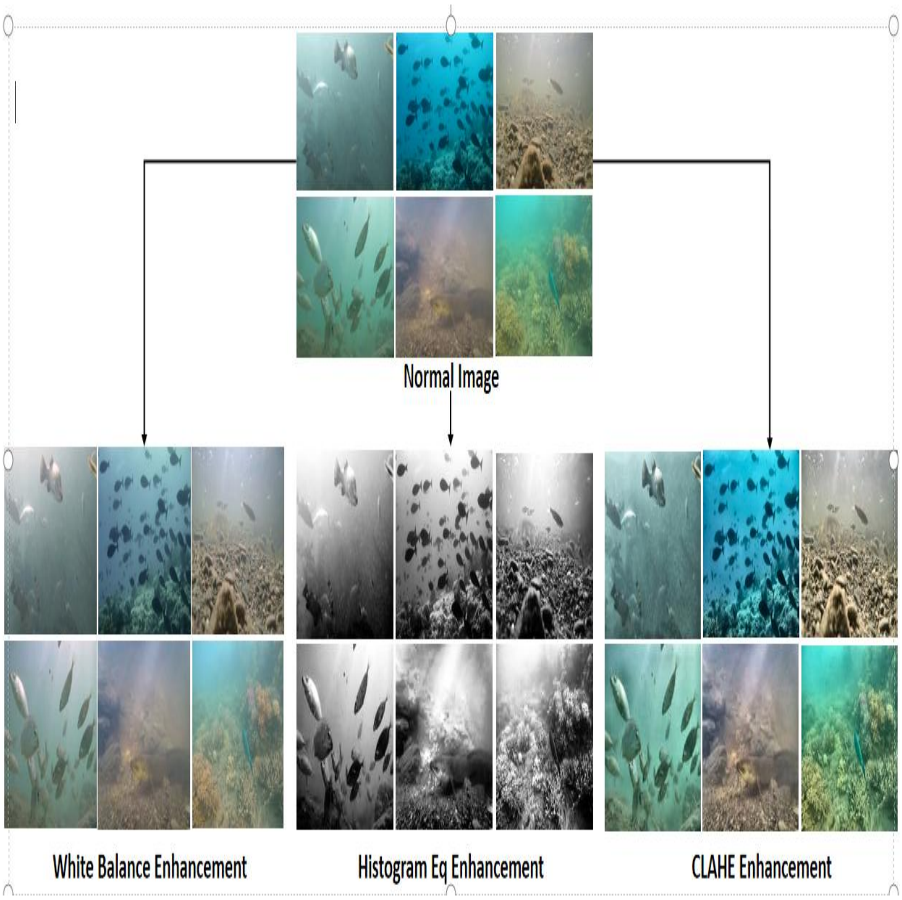
- Corrects for the colour distortion in underwater images caused by light attenuation and scattering.
- Corrects the RGB (Red, Green, Blue) channels of an image and Uses the green channel as a reference point as it is the least impacted by the absorption properties of water.
- Improves image quality to make fish edges and features more visible.

3.3.2. Histogram Equalization: -

- Brightness and contrast in underwater images are increased in order to help efficiently identify fish.
- tool operates on the HSV (Hue, Saturation, Value) color space and uses the Value (brightness) layer only.
- Essentially clarified fish boundaries to improve the feature extraction for detection models.
  Used a cumulative distribution function (CDF) to reshape pixel intensity values.

3.3.3. Clahe: -

- It stands for Contrast Limited Adaptive Histogram Equalization.
- Divides the image into small 8x8 tile grids for localized processing and it Applies adaptive histogram equalization to each tile, redistributing pixel intensities.
- Uses a clip limit of 2.0 to prevent noise amplification and over-enhancement.
- Enhances detection accuracy and it Improves visibility of fine details (e.g., fish textures) in varying underwater conditions.



➢<u>Hybrid Enhancement Pipeline Images:-</u>

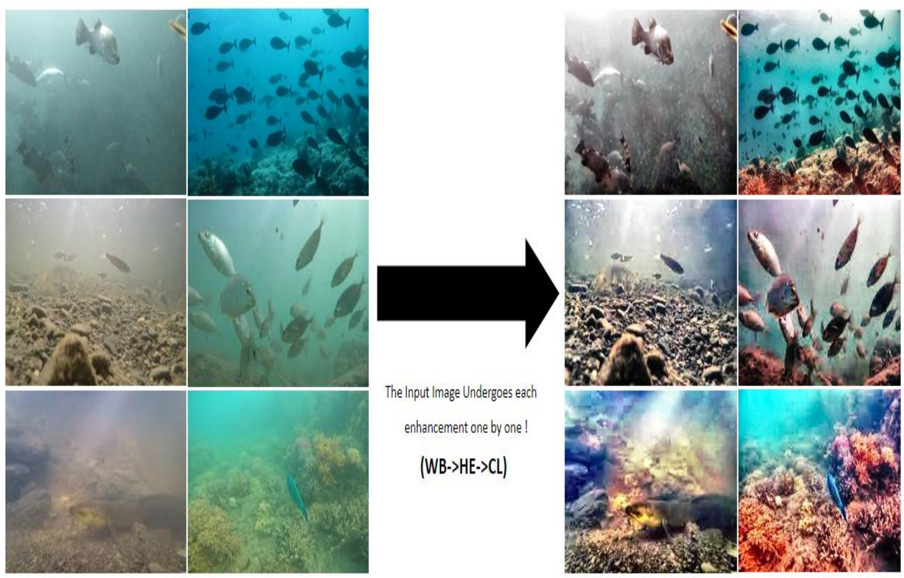


Figure 3.3 :- Normal images to Enhanced Images

3.4. Code Implementation: -

3.4.1. FasterRCNN working model code: -

```
[ ] import torch
    import torchvision
    from torch.utils.data import DataLoader
    from torchvision.models.detection import FasterRCNN
    from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
    from torchvision.datasets import CocoDetection
    from torchvision.transforms import functional as F
    import matplotlib.pyplot as plt
    from PIL import Image
```

```
# Define transformations
class CocoTransform:
    def __call__(self, image, target):
        image = F.to_tensor(image)  # Convert PIL image to tensor
        return image, target
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] # Dataset class
    def get_coco_dataset(img_dir, ann_file):
        return CocoDetection(
            root=img_dir,
            annFile=ann_file,
            transforms=CocoTransform()
        )

    # Load datasets
    train_dataset = get_coco_dataset(
        img_dir="/content/drive/MyDrive/My First Project.v3i.coco (1)/train",
        ann_file="/content/drive/MyDrive/My First Project.v3i.coco (1)/train/_annotations.coco.json"
    )


    val_dataset = get_coco_dataset(
        img_dir="/content/drive/MyDrive/My First Project.v3i.coco (1)/valid",
        ann_file="/content/drive/MyDrive/My First Project.v3i.coco (1)/valid/_annotations.coco.json"
    )
    # DataLoader
    train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True, collate_fn=lambda x: tuple(zip(*x)))
    val_loader = DataLoader(val_dataset, batch_size=4, shuffle=False, collate_fn=lambda x: tuple(zip(*x)))
```

```
loading annotations into memory...
Done (t=10.27s)
creating index...
index created!
loading annotations into memory...
Done (t=0.57s)
creating index...
index created!
```

```
[ ] # Load Faster R-CNN with ResNet-50 backbone
    def get_model(num_classes):
        # Load pre-trained Faster R-CNN
        model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

        # Get the number of input features for the classifier
        in_features = model.roi_heads.box_predictor.cls_score.in_features

        # Replace the pre-trained head with a new one
        model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
        return model
```

```python
# Load Faster R-CNN with ResNet-50 backbone
def get_model(num_classes):
    # Load pre-trained Faster R-CNN
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

    # Get the number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features

    # Replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
    return model
```

```python
# Initialize the model
num_classes = 6 # Background + chair, human, table
model = get_model(num_classes)
```

```
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth" to /root/.cache/torch/hub/checkpoints/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth
100%|██████████| 160M/160M [00:01<00:00, 127MB/s]
```

```python
# Move model to GPU if available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model.to(device)

# Define optimizer and learning rate scheduler
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)
```

```
[ ] def train_one_epoch(model, optimizer, data_loader, device, epoch):
        model.train()
        for images, targets in data_loader:
            # Move images to the device
            images = [img.to(device) for img in images]

            # Validate and process targets
            processed_targets = []
            valid_images = []
            for i, target in enumerate(targets):
                boxes = []
                labels = []
                for obj in target:
                    # Extract bbox
                    bbox = obj["bbox"]  # Format: [x, y, width, height]
                    x, y, w, h = bbox

                    # Ensure the width and height are positive
                    if w > 0 and h > 0:
                        boxes.append([x, y, x + w, y + h])  # Convert to [x_min, y_min, x_max, y_max]
                        labels.append(obj["category_id"])

                # Only process if there are valid boxes
                if boxes:
                    processed_target = {
                        "boxes": torch.tensor(boxes, dtype=torch.float32).to(device),
                        "labels": torch.tensor(labels, dtype=torch.int64).to(device),
                    }
                    processed_targets.append(processed_target)
                    valid_images.append(images[i])  # Add only valid images

            # Skip iteration if no valid targets
            if not processed_targets:
                continue

            # Ensure images and targets are aligned
            images = valid_images

            # Forward pass
            loss_dict = model(images, processed_targets)
            losses = sum(loss for loss in loss_dict.values())

            # Backpropagation
            optimizer.zero_grad()
            losses.backward()
            optimizer.step()

        print(f"Epoch [{epoch}] Loss: {losses.item():.4f}")
```

```python
# Training loop
num_epochs = 25
for epoch in range(num_epochs):
    train_one_epoch(model, optimizer, train_loader, device, epoch)
    lr_scheduler.step()

    # Save the model's state dictionary after every epoch
    model_path = f"fasterrcnn_resnet50_epoch_{epoch + 1}.pth"
    torch.save(model.state_dict(), model_path)
    print(f"Model saved: {model_path}")
```

```python
import torch
import torchvision
from torch.utils.data import DataLoader
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.transforms import functional as F
import matplotlib.pyplot as plt
from PIL import Image

# Load Faster R-CNN with ResNet-50 backbone
def get_model(num_classes):
    # Load pre-trained Faster R-CNN
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(weights="DEFAULT")
    # Get the number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # Replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
    return model


# Initialize the model
num_classes = 6  # Background + small + medium + large

# Move model to GPU if available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')

# Load the trained model
model = get_model(num_classes)
model.load_state_dict(torch.load("fasterrcnn_resnet50_epoch_18.pth", map_location=device, weights_only=True))
model.to(device)
model.eval()  # Set the model to evaluation mode


# Function to preprocess image
def prepare_image(image_path):
    image = Image.open(image_path).convert("RGB")  # Open image
    image_tensor = F.to_tensor(image).unsqueeze(0)  # Convert image to tensor and add batch dimension
    return image_tensor.to(device)

# Define class names
# COCO_CLASSES = {0: "Background", 1: "yellofish", 2: "bluefish", 3: "goldfish"}
COCO_CLASSES = {0: "Background", 1: "Bluetang", 2: "MoorishIdol", 3: "Nemo", 4: "YellowTang", 5: "Goldfish"}

def get_class_name(class_id):
    return COCO_CLASSES.get(class_id, "Unknown")
```

```python
# Function to draw bounding boxes
def draw_boxes(image, prediction, fig_size=(12, 10)):
    plt.figure(figsize=fig_size)  # Set figure size
    plt.imshow(image)  # Display the image first

    boxes = prediction[0]['boxes'].cpu().numpy()  # Get bounding boxes
    labels = prediction[0]['labels'].cpu().numpy()  # Get labels
    scores = prediction[0]['scores'].cpu().numpy()  # Get confidence scores

    threshold = 0.5  # Lowered threshold for testing

    print("Boxes Shape:", boxes.shape)
    print("Labels Shape:", labels.shape)
    print("Scores Shape:", scores.shape)

    for box, label, score in zip(boxes, labels, scores):
        if score > threshold:
            x_min, y_min, x_max, y_max = box
            class_name = get_class_name(label)  # Get class name

            # Ensure coordinates are within image bounds
            x_min = max(0, x_min)
            y_min = max(0, y_min)
            x_max = min(image.size[0], x_max)
            y_max = min(image.size[1], y_max)

            # Draw bounding box
            plt.gca().add_patch(plt.Rectangle(
                (x_min, y_min), x_max - x_min, y_max - y_min,
                linewidth=2, edgecolor='r', facecolor='none'
            ))

            # Add text label
            plt.text(x_min, y_min, f"{class_name} ({score:.2f})", color='r', fontsize=12,
                     bbox=dict(facecolor='white', alpha=0.5))

    plt.axis('off')  # Hide axes
    plt.show()


# Load the unseen image
image_path = "/content/drive/MyDrive/coco/test/7117_Caranx_sexfasciatus_juvenile_f000022_RGHS_jpg.rf.2b7c2322f58bf67d49e96446c2b2e560.jpg"
image_tensor = prepare_image(image_path)

# Run inference
with torch.no_grad():  # Disable gradient computation
    prediction = model(image_tensor)

# Display results
draw_boxes(Image.open(image_path), prediction)
```

3.4.2. Yolov11 working model code: -

```
%pip install ultralytics
import ultralytics
ultralytics.checks()
```

```
Ultralytics 8.3.100 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 41.2/112.6 GB disk)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Download COCO val
import torch
torch.hub.download_url_to_file('https://ultralytics.com/assets/coco2017val.zip', 'tmp.zip')  # download (780M - 5000 images)
!unzip -q tmp.zip -d datasets && rm tmp.zip  # unzip
```

```
100%|██████████| 780M/780M [00:22<00:00, 35.8MB/s]
```

```
# Validate YOLO11n on COCO8 val
!yolo val model=yolo11n.pt data='/content/drive/MyDrive/MIT/augmented.yaml'
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11n.pt to 'yolo11n.pt'...
100% 5.35M/5.35M [00:00<00:00, 276MB/s]
Ultralytics 8.3.100 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
YOLO11n summary (fused): 100 layers, 2,616,248 parameters, 0 gradients, 6.5 GFLOPs
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 63.6MB/s]
val: Scanning /content/drive/MyDrive/MIT/labels/val... 52 images, 0 backgrounds, 0 corrupt: 100% 52/52 [01:14<00:00,  1.43s/it]
val: New cache created: /content/drive/MyDrive/MIT/labels/val.cache
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:05<00:00,  1.30s/it]
                   all         52       1635          0          0          0          0
                person         52       1141          0          0          0          0
               bicycle         49        494          0          0          0          0
Speed: 6.1ms preprocess, 28.6ms inference, 0.0ms loss, 11.8ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

```
# Train YOLO11n on COCO8 for 3 epochs
!yolo train model=yolo11n.pt data='/content/drive/MyDrive/MIT/augmented.yaml' epochs=500 imgsz=640
```

```
Ultralytics 8.3.100 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: task=detect, mode=train, model=yolo11n.pt, data=/content/drive/MyDrive/MIT/augmented.yaml, epochs=500, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1743567306.852534   13925 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1743567306.859388   13925 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
Overriding model.yaml nc=80 with nc=2

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      6640  ultralytics.nn.modules.block.C3k2            [32, 64, 1, False, 0.25]
  3                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
  4                  -1  1     26080  ultralytics.nn.modules.block.C3k2            [64, 128, 1, False, 0.25]
  5                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
  6                  -1  1     87040  ultralytics.nn.modules.block.C3k2            [128, 128, 1, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    346112  ultralytics.nn.modules.block.C3k2            [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1    249728  ultralytics.nn.modules.block.C2PSA           [256, 256, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 12             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 13                  -1  1    111296  ultralytics.nn.modules.block.C3k2            [384, 128, 1, False]
 14                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 15             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 16                  -1  1     32096  ultralytics.nn.modules.block.C3k2            [256, 64, 1, False]
 17                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 18            [-1, 13]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 19                  -1  1     86720  ultralytics.nn.modules.block.C3k2            [192, 128, 1, False]
 20                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 21            [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 22                  -1  1    378880  ultralytics.nn.modules.block.C3k2            [384, 256, 1, True]
 23        [16, 19, 22]  1    431062  ultralytics.nn.modules.head.Detect           [2, [64, 128, 256]]
YOLO11n summary: 181 layers, 2,590,230 parameters, 2,590,214 gradients, 6.4 GFLOPs

Transferred 448/499 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train2', view at http://localhost:6006/
Freezing layer 'model.23.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Scanning /content/drive/MyDrive/MIT/labels/train.cache... 206 images, 0 backgrounds, 0 corrupt: 100% 206/206 [00:00<?, ?it/s]
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8))
val: Scanning /content/drive/MyDrive/MIT/labels/val.cache... 52 images, 0 backgrounds, 0 corrupt: 100% 52/52 [00:00<?, ?it/s]
Plotting labels to runs/detect/train2/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.001667, momentum=0.9) with parameter groups 81 weight(decay=0.0), 88 weight(decay=0.0005), 87 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train2
Starting training for 500 epochs...
```

```
Transferred 448/499 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train2', view at http://localhost:6006/
Freezing layer 'model.23.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Scanning /content/drive/MyDrive/MIT/labels/train.cache... 206 images, 0 backgrounds, 0 corrupt: 100% 206/206 [00:00<?, ?it/s]
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8))
val: Scanning /content/drive/MyDrive/MIT/labels/val.cache... 52 images, 0 backgrounds, 0 corrupt: 100% 52/52 [00:00<?, ?it/s]
Plotting labels to runs/detect/train2/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.001667, momentum=0.9) with parameter groups 81 weight(decay=0.0), 88 weight(decay=0.0005), 87 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train2
Starting training for 500 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/500      3.19G      2.161      3.634      1.075        681        640: 100% 13/13 [00:11<00:00,  1.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:01<00:00,  1.27it/s]
                   all         52       1635     0.0303      0.257     0.0255     0.0111

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
[ ] # Run inference on an image with YOLO11n
    !yolo predict model='/content/runs/detect/train5/weights/best.pt' source= '/content/drive/MyDrive/yolo/test/images/7117_Caranx_sexfasciatus_juvenile_f000003_RGHS_jpg.rf.c41ade83491275abda255248ac837fe3.jpg'
```

```
Ultralytics 8.3.97 🚀 Python-3.11.11 torch-2.6.0+cu124 CPU (Intel Xeon 2.20GHz)
YOLO11n summary (fused): 100 layers, 2,583,127 parameters, 0 gradients, 6.3 GFLOPs

image 1/1 /content/drive/MyDrive/yolo/test/images/7117_Caranx_sexfasciatus_juvenile_f000003_RGHS_jpg.rf.c41ade83491275abda255248ac837fe3.jpg: 640x640 2 caranxs, 406.9ms
Speed: 11.5ms preprocess, 406.9ms inference, 7.6ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

In this project, we used fish detection by applying two advanced deep learning models to provide robust fish identification. The two models we used were YOLOv11 and Faster R-CNN. Before we fed the images into our models, we made use of image enhancement capabilities highlighted by image enhancement processes beyond the data transformation model, such as histogram equalization and contrast. YOLOv11 was used in a real-time detection mode due to its speed and efficiency while Faster R-CNN was used for its strong helpfulness in identifying fish species, localizing at the same time. Both models were re-tuned on the image enhanced data, and incorporated data augmentation techniques. The results showed improvement in fish detection rates with both models operating on different datasets. The novelty of this project is to highlight a use of YOLOv11 and Faster R-CNN for stakeholder (identifying and monitoring fish) with a level of precision worthy of attention across various aquatic environments.

3.4.3. Fish Detection Application code:-

```
[ ]: import torch
import tkinter as tk
from tkinter import filedialog, Label, Button, Canvas, Text, messagebox
import cv2
from PIL import Image, ImageTk
from torchvision import models, transforms
from ultralytics import YOLO
import numpy as np
import os
import time
import random

class FishDetectionGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Fish Detection GUI")
        self.root.geometry("1000x800")
        self.center_window()

        # Heading and Logo (Centered at Top)
        self.heading_frame = tk.Frame(root)
        self.heading_frame.pack(pady=20)

        self.college_logo = Image.open("D:/Major_Project/srmlogo.png")
        self.college_logo = self.college_logo.resize((400, 100))
        self.college_logo_image = ImageTk.PhotoImage(self.college_logo)
        self.logo_label = Label(self.heading_frame, image=self.college_logo_image)
        self.logo_label.pack(side=tk.LEFT, padx=10)

        self.heading = Label(self.heading_frame, text="Fish Detection GUI", font=("Arial", 20, "bold"))
        self.heading.pack(side=tk.LEFT)

        # Main Frame for Layout
        self.main_frame = tk.Frame(root)
        self.main_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        # Left Side: Canvas for Display with Title
        self.canvas_frame = tk.Frame(self.main_frame)
        self.canvas_frame.pack(side=tk.LEFT, padx=(10, 2))
```

```
self.canvas_frame = tk.Frame(self.main_frame)
self.canvas_frame.pack(side=tk.LEFT, padx=(10, 2))

self.canvas_title = Label(self.canvas_frame, text="Display Screen", font=("Arial", 14, "bold"))
self.canvas_title.pack(pady=(0, 5))

self.canvas = Canvas(self.canvas_frame, width=600, height=400, bg="gray", highlightthickness=1, highlightbackground="black")
self.canvas.pack()

# Right Side: Buttons in a Box
self.button_frame = tk.Frame(self.main_frame, borderwidth=2, relief="groove")
self.button_frame.pack(side=tk.RIGHT, padx=(2, 10), fill=tk.Y)

self.button_title = Label(self.button_frame, text="Controls", font=("Arial", 14, "bold"))
self.button_title.pack(pady=(10, 5))

self.label = Label(self.button_frame, text="Select Detection Method:", font=("Arial", 10))
self.label.pack(pady=(5, 2))
self.detection_method = tk.StringVar(value="yolo")
tk.Radiobutton(self.button_frame, text="YOLOv11", variable=self.detection_method, value="yolo", font=("Arial", 10)).pack(anchor=tk.W, padx=10)
tk.Radiobutton(self.button_frame, text="Faster R-CNN", variable=self.detection_method, value="fasterrcnn", font=("Arial", 10)).pack(anchor=tk.W,
tk.Radiobutton(self.button_frame, text="Haar Cascade", variable=self.detection_method, value="haar", font=("Arial", 10)).pack(anchor=tk.W, padx=1

self.upload_button = Button(self.button_frame, text="Upload Image/Video", command=self.upload_image_or_video, font=("Arial", 10))
self.upload_button.pack(pady=10, padx=10, fill=tk.X)

self.enhance_button = Button(self.button_frame, text="Enhance Image/Video", command=self.enhance_image_or_video, state="disabled", font=("Arial",
self.enhance_button.pack(pady=10, padx=10, fill=tk.X)

self.detect_button = Button(self.button_frame, text="Detect Fish", command=self.detect_fish, state="disabled", font=("Arial", 10))
self.detect_button.pack(pady=10, padx=10, fill=tk.X)

self.download_image_button = Button(self.button_frame, text="Download Image", command=self.download_image, state="disabled", font=("Arial", 10))
self.download_image_button.pack(pady=10, padx=10, fill=tk.X)

self.download_video_button = Button(self.button_frame, text="Download Video", command=self.download_video, state="disabled", font=("Arial", 10))
self.download_video_button.pack(pady=10, padx=10, fill=tk.X)

# Separator Line
self.separator = tk.Frame(root, height=2, bd=1, relief="sunken")
self.separator.pack(fill=tk.X, padx=10, pady=5)
```

```
self.label = Label(self.button_frame, text="Select Detection Method:", font=("Arial", 10))
self.label.pack(pady=(5, 2))
self.detection_method = tk.StringVar(value="yolo")
tk.Radiobutton(self.button_frame, text="YOLOv11", variable=self.detection_method, value="yolo", font=("Arial", 10)).pack(anchor=tk.W, padx=10)
tk.Radiobutton(self.button_frame, text="Faster R-CNN", variable=self.detection_method, value="fasterrcnn", font=("Arial", 10)).pack(anchor=tk.W,
tk.Radiobutton(self.button_frame, text="Haar Cascade", variable=self.detection_method, value="haar", font=("Arial", 10)).pack(anchor=tk.W, padx=1

self.upload_button = Button(self.button_frame, text="Upload Image/Video", command=self.upload_image_or_video, font=("Arial", 10))
self.upload_button.pack(pady=10, padx=10, fill=tk.X)

self.enhance_button = Button(self.button_frame, text="Enhance Image/Video", command=self.enhance_image_or_video, state="disabled", font=("Arial",
self.enhance_button.pack(pady=10, padx=10, fill=tk.X)

self.detect_button = Button(self.button_frame, text="Detect Fish", command=self.detect_fish, state="disabled", font=("Arial", 10))
self.detect_button.pack(pady=10, padx=10, fill=tk.X)

self.download_image_button = Button(self.button_frame, text="Download Image", command=self.download_image, state="disabled", font=("Arial", 10))
self.download_image_button.pack(pady=10, padx=10, fill=tk.X)

self.download_video_button = Button(self.button_frame, text="Download Video", command=self.download_video, state="disabled", font=("Arial", 10))
self.download_video_button.pack(pady=10, padx=10, fill=tk.X)

# Separator Line
self.separator = tk.Frame(root, height=2, bd=1, relief="sunken")
self.separator.pack(fill=tk.X, padx=10, pady=5)

# Bottom: Results
self.results_frame = tk.Frame(root)
self.results_frame.pack(fill=tk.X, padx=10, pady=10)

self.results_label = Label(self.results_frame, text="Results:", font=("Arial", 12, "bold"))
self.results_label.pack()
self.results_text = Text(self.results_frame, height=10, width=80, font=("Arial", 10))
self.results_text.pack()

# Fish Classes
self.class_names = ["Epinephelus", "Chaetodon_Vagabundus", "Caranx", "Gerres", "Acanthopagrus"]

# Hardcoded Model Paths
yolo_paths = [
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/YoloV11/internetimages.pt",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/YoloV11/mangrooveforest.pt"  # Replace with your second YOLO path
]
```

```python
fasterrcnn_paths = [
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/FasterRCNN/internetimages.pth",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/FasterRCNN/mangrooveforest.pth"  # Replace with your second Faster R-CNN
]
haar_paths = [
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/HaarCascade/Acanthopagrus_Palmaris.xml",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/HaarCascade/Caranx.xml",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/HaarCascade/Chaetodon_Vagabundus.xml",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/HaarCascade/Epinephelus.xml",
    "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Dataset/HaarCascade/Gerres.xml"
]

# Load Models
self.yolo_models = [None, None]
self.fasterrcnn_models = [None, None]
try:
    self.yolo_models[0] = YOLO(yolo_paths[0])
    self.yolo_models[1] = YOLO(yolo_paths[1])
except Exception as e:
    messagebox.showerror("Error", f"Failed to load YOLO models: {str(e)}")
try:
    num_classes = 6
    for i, path in enumerate(fasterrcnn_paths):
        model = models.detection.fasterrcnn_resnet50_fpn(pretrained=False, num_classes=num_classes)
        model.load_state_dict(torch.load(path, map_location=torch.device('cpu')))
        model.eval()
        self.fasterrcnn_models[i] = model
except Exception as e:
    messagebox.showerror("Error", f"Failed to load Faster R-CNN models: {str(e)}")
self.haar_cascade = {
    "Acanthopagrus Palmaris": haar_paths[0],
    "Caranx": haar_paths[1],
    "Chaetodon Vagabundus": haar_paths[2],
    "Epinephelus": haar_paths[3],
    "Gerres": haar_paths[4]
}
```

```
        # Initialize variables
        self.file_path = None
        self.is_video = False
        self.original_image = None
        self.enhanced_image = None
        self.current_display = None
        self.download_path = "C:/Users/Wilfred Auxilian/Desktop/Conference_Files/GUI_No3/Downloaded_Images_Videos"
        self.detected_frames = []
        self.detected_video_writer = None
        self.playing = False
        self.all_detected_fish = set()
        self.current_method = None

    def center_window(self):
        window_width = 1000
        window_height = 800
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()
        position_top = int(screen_height / 2 - window_height / 2)
        position_left = int(screen_width / 2 - window_width / 2)
        self.root.geometry(f'{window_width}x{window_height}+{position_left}+{position_top}')

    def upload_image_or_video(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.png;*.jpeg"), ("Video Files", "*.mp4;*.avi;*.mov")])
        if file_path:
            self.file_path = file_path
            self.is_video = file_path.endswith(('.mp4', '.avi', '.mov'))
            self.enhance_button.config(state="normal")
            self.detect_button.config(state="normal")
            self.download_image_button.config(state="normal")
            self.download_video_button.config(state="normal" if self.is_video else "disabled")
            self.detected_frames = []
            self.all_detected_fish = set()
            self.playing = False
            self.results_text.delete(1.0, tk.END)
            self.display_image_or_video()
```

```python
def display_image_or_video(self):
    if self.is_video:
        cap = cv2.VideoCapture(self.file_path)
        ret, frame = cap.read()
        if ret:
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image = Image.fromarray(frame_rgb).resize((600, 400))
            self.tk_image = ImageTk.PhotoImage(image)
            self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
            self.canvas.image = self.tk_image
            self.current_display = frame
        cap.release()
    else:
        self.original_image = cv2.imread(self.file_path)
        image_rgb = cv2.cvtColor(self.original_image, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(image_rgb).resize((600, 400))
        self.tk_image = ImageTk.PhotoImage(image)
        self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
        self.canvas.image = self.tk_image
        self.current_display = self.original_image

def enhance_image_or_video(self):
    if self.is_video:
        self.enhance_video()
    else:
        self.enhance_image()
    self.detect_button.config(state="normal")
    self.download_image_button.config(state="normal")
    self.download_video_button.config(state="normal" if self.is_video else "disabled")
```

```python
def enhance_image(self):
    if self.original_image is None:
        return

    b, g, r = cv2.split(self.original_image)
    b_mean, g_mean, r_mean = np.mean(b), np.mean(g), np.mean(r)
    b_scale, r_scale = g_mean / (b_mean + 1e-6), g_mean / (r_mean + 1e-6)
    b_wb = np.clip(b.astype(np.float32) * b_scale, 0, 255).astype(np.uint8)
    g_wb = g
    r_wb = np.clip(r.astype(np.float32) * r_scale, 0, 255).astype(np.uint8)
    white_balanced = cv2.merge([b_wb, g_wb, r_wb])

    hsv_eq = cv2.cvtColor(white_balanced, cv2.COLOR_BGR2HSV)
    h_eq, s_eq, v_eq = cv2.split(hsv_eq)
    v_equalized = cv2.equalizeHist(v_eq)
    hsv_equalized = cv2.merge([h_eq, s_eq, v_equalized])
    color_histogram_eq = cv2.cvtColor(hsv_equalized, cv2.COLOR_HSV2BGR)

    hsv_clahe = cv2.cvtColor(color_histogram_eq, cv2.COLOR_BGR2HSV)
    h_clahe, s_clahe, v_clahe = cv2.split(hsv_clahe)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    v_clahe_enhanced = clahe.apply(v_clahe)
    hsv_clahe_enhanced = cv2.merge([h_clahe, s_clahe, v_clahe_enhanced])
    self.enhanced_image = cv2.cvtColor(hsv_clahe_enhanced, cv2.COLOR_HSV2BGR)

    enhanced_rgb = cv2.cvtColor(self.enhanced_image, cv2.COLOR_BGR2RGB)
    enhanced_pil = Image.fromarray(enhanced_rgb).resize((600, 400))
    self.tk_image = ImageTk.PhotoImage(enhanced_pil)
    self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
    self.canvas.image = self.tk_image
    self.current_display = self.enhanced_image
```

```python
def enhance_video(self):
    cap = cv2.VideoCapture(self.file_path)
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter("enhanced_video.mp4", fourcc, 20.0, (int(cap.get(3)), int(cap.get(4))))

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        b, g, r = cv2.split(frame)
        b_mean, g_mean, r_mean = np.mean(b), np.mean(g), np.mean(r)
        b_scale, r_scale = g_mean / (b_mean + 1e-6), g_mean / (r_mean + 1e-6)
        b_wb = np.clip(b.astype(np.float32) * b_scale, 0, 255).astype(np.uint8)
        g_wb = g
        r_wb = np.clip(r.astype(np.float32) * r_scale, 0, 255).astype(np.uint8)
        white_balanced = cv2.merge([b_wb, g_wb, r_wb])

        hsv_eq = cv2.cvtColor(white_balanced, cv2.COLOR_BGR2HSV)
        h_eq, s_eq, v_eq = cv2.split(hsv_eq)
        v_equalized = cv2.equalizeHist(v_eq)
        hsv_equalized = cv2.merge([h_eq, s_eq, v_equalized])
        color_histogram_eq = cv2.cvtColor(hsv_equalized, cv2.COLOR_HSV2BGR)

        hsv_clahe = cv2.cvtColor(color_histogram_eq, cv2.COLOR_BGR2HSV)
        h_clahe, s_clahe, v_clahe = cv2.split(hsv_clahe)
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
        v_clahe_enhanced = clahe.apply(v_clahe)
        hsv_clahe_enhanced = cv2.merge([h_clahe, s_clahe, v_clahe_enhanced])
        enhanced_frame = cv2.cvtColor(hsv_clahe_enhanced, cv2.COLOR_HSV2BGR)

        out.write(enhanced_frame)
        frame_rgb = cv2.cvtColor(enhanced_frame, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(frame_rgb).resize((600, 400))
        self.tk_image = ImageTk.PhotoImage(image)
        self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
        self.canvas.image = self.tk_image
        self.root.update()

    cap.release()
    out.release()
    self.file_path = "enhanced_video.mp4"
```

```python
def detect_fish(self):
    method = self.detection_method.get()
    self.detected_frames = []
    self.all_detected_fish = set()
    self.current_method = method
    self.playing = True

    # Validate model availability
    if method == "yolo" and (not self.yolo_models[0] or not self.yolo_models[1]):
        messagebox.showerror("Error", "One or both YOLO models are not loaded!")
        return
    elif method == "fasterrcnn" and (not self.fasterrcnn_models[0] or not self.fasterrcnn_models[1]):
        messagebox.showerror("Error", "One or both Faster R-CNN models are not loaded!")
        return

    if self.is_video:
        self.detect_video(method)
    else:
        if self.enhanced_image is not None:
            cv2.imwrite("temp_enhanced.jpg", self.enhanced_image)
            input_path = "temp_enhanced.jpg"
        else:
            input_path = self.file_path

        detected_fish = []
        if method == "yolo":
            detected_fish = self.detect_with_yolo(input_path)
        elif method == "fasterrcnn":
            detected_fish = self.detect_with_fasterrcnn(input_path)
        elif method == "haar":
            detected_fish = self.detect_with_haar(input_path)

        self.update_results(detected_fish, method)
        if not detected_fish:
            messagebox.showinfo("Detection Result", "No fish detected in the image. Please Train Your Model More!")
```

```python
def detect_video(self, method):
    cap = cv2.VideoCapture(self.file_path)
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_delay = int(1000 / fps)
    frame_count = 0
    skip_frames = 5

    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    self.detected_video_writer = cv2.VideoWriter("temp_detected_video.mp4", fourcc, fps, (int(cap.get(3)), int(cap.get(4))))

    while cap.isOpened() and self.playing:
        ret, frame = cap.read()
        if not ret:
            break

        frame_small = cv2.resize(frame, (int(frame.shape[1] * 0.5), int(frame.shape[0] * 0.5)))

        if frame_count % skip_frames == 0:
            temp_frame_path = "temp_frame.jpg"
            cv2.imwrite(temp_frame_path, frame_small)

            detected_fish = []
            if method == "yolo":
                detected_fish = self.detect_with_yolo_frame(temp_frame_path, frame_small)
            elif method == "fasterrcnn":
                detected_fish = self.detect_with_fasterrcnn_frame(temp_frame_path, frame_small)
            elif method == "haar":
                detected_fish = self.detect_with_haar_frame(temp_frame_path, frame_small)

            self.all_detected_fish.update(detected_fish)
            self.update_results(list(self.all_detected_fish), method)

            frame_with_detections = cv2.imread(temp_frame_path)
            frame_with_detections = cv2.resize(frame_with_detections, (frame.shape[1], frame.shape[0]))
        else:
            frame_with_detections = frame
```

```
            frame_rgb = cv2.cvtColor(frame_with_detections, cv2.COLOR_BGR2RGB)
            image = Image.fromarray(frame_rgb).resize((600, 400))
            self.tk_image = ImageTk.PhotoImage(image)
            self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
            self.canvas.image = self.tk_image
            self.current_display = frame_with_detections

            self.root.update()
            self.root.after(frame_delay)

            frame_count += 1

        cap.release()
        self.detected_video_writer.release()
        if os.path.exists("temp_frame.jpg"):
            os.remove("temp_frame.jpg")
        self.playing = False

        self.root.update()
        if not self.all_detected_fish:
            messagebox.showinfo("Detection Result", "No fish detected in the video. Please Train Your Model More!")

    def detect_with_yolo_frame(self, input_path, frame):
        yolo_class_names = ["Chaetodon_Vagabundus", "Epinephelus", "Acanthopagrus", "Caranx", "Gerres"]
        detected_fish = []
        image_cv = cv2.imread(input_path)

        # Run both YOLO models
        for model in self.yolo_models:
            results = model(input_path)
            for pred in results[0].boxes:
                fish_class = pred.cls
                if fish_class != 0:
                    class_name = yolo_class_names[int(fish_class) - 1]
                    detected_fish.append(class_name)
                    # Draw bounding box
                    xyxy = pred.xyxy[0].cpu().numpy()
                    x1, y1, x2, y2 = map(int, xyxy)
                    cv2.rectangle(image_cv, (x1, y1), (x2, y2), (0, 255, 0), 2)
                    cv2.putText(image_cv, class_name, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
```

```
    cv2.imwrite(input_path, image_cv)
    if detected_fish:
        self.detected_frames.append(frame)
    return list(set(detected_fish))

def detect_with_fasterrcnn_frame(self, input_path, frame):
    image = Image.open(input_path).convert("RGB")
    transform = transforms.ToTensor()
    image_tensor = transform(image).unsqueeze(0)
    image_cv = cv2.imread(input_path)
    detected_fish = []

    # Run both Faster R-CNN models
    for model in self.fasterrcnn_models:
        with torch.no_grad():
            predictions = model(image_tensor)
        boxes = predictions[0]['boxes']
        labels = predictions[0]['labels']
        scores = predictions[0]['scores']
        for i, score in enumerate(scores):
            if score > 0.5:
                x1, y1, x2, y2 = boxes[i].cpu().numpy()
                label_idx = labels[i].item()
                if label_idx > 0:
                    label_name = self.class_names[label_idx - 1]
                    detected_fish.append(label_name)
                    cv2.rectangle(image_cv, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
                    cv2.putText(image_cv, label_name, (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    cv2.imwrite(input_path, image_cv)
    if detected_fish:
        self.detected_frames.append(frame)
    return list(set(detected_fish))
```

```
def detect_with_haar_frame(self, input_path, frame):
    image = cv2.imread(input_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    detected_fish = []
    for fish_name, cascade_path in self.haar_cascade.items():
        fish_cascade = cv2.CascadeClassifier(cascade_path)
        fish = fish_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in fish:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            detected_fish.append(fish_name)
            cv2.putText(image, fish_name, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
    cv2.imwrite(input_path, image)
    if detected_fish:
        self.detected_frames.append(frame)
    return detected_fish

def detect_with_yolo(self, input_path):
    yolo_class_names = ["Chaetodon_Vagabundus", "Epinephelus", "Acanthopagrus", "Caranx", "Gerres"]
    detected_fish = []
    image_cv = cv2.imread(input_path)

    # Run both YOLO models
    for model in self.yolo_models:
        results = model(input_path)
        for pred in results[0].boxes:
            fish_class = pred.cls
            if fish_class != 0:
                class_name = yolo_class_names[int(fish_class) - 1]
                detected_fish.append(class_name)
                # Draw bounding box
                xyxy = pred.xyxy[0].cpu().numpy()
                x1, y1, x2, y2 = map(int, xyxy)
                cv2.rectangle(image_cv, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(image_cv, class_name, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    output_path = "output_yolo.jpg"
    cv2.imwrite(output_path, image_cv)
    self.display_result(output_path)
    return list(set(detected_fish))
```

```python
def detect_with_fasterrcnn(self, input_path):
    image = Image.open(input_path).convert("RGB")
    transform = transforms.ToTensor()
    image_tensor = transform(image).unsqueeze(0)
    image_cv = cv2.imread(input_path)
    detected_fish = []

    # Run both Faster R-CNN models
    for model in self.fasterrcnn_models:
        with torch.no_grad():
            predictions = model(image_tensor)
        boxes = predictions[0]['boxes']
        labels = predictions[0]['labels']
        scores = predictions[0]['scores']
        for i, score in enumerate(scores):
            if score > 0.5:
                x1, y1, x2, y2 = boxes[i].cpu().numpy()
                label_idx = labels[i].item()
                if label_idx > 0:
                    label_name = self.class_names[label_idx - 1]
                    detected_fish.append(label_name)
                    cv2.rectangle(image_cv, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
                    cv2.putText(image_cv, label_name, (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    output_path = "fasterrcnn_output.jpg"
    cv2.imwrite(output_path, image_cv)
    self.display_result(output_path)
    return list(set(detected_fish))

def detect_with_haar(self, input_path):
    image = cv2.imread(input_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    detected_fish = []
    for fish_name, cascade_path in self.haar_cascade.items():
        fish_cascade = cv2.CascadeClassifier(cascade_path)
        fish = fish_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in fish:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            detected_fish.append(fish_name)
            cv2.putText(image, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
    cv2.imwrite("haar_output.jpg", image)
    self.display_result("haar_output.jpg")
    return detected_fish
```

```
def estimate_distance_and_area(self, frame):
    height, width = frame.shape[:2]
    focal_length = 1000
    object_size = 1.0
    distance = (object_size * focal_length) / max(height, width) * 1000
    distance = distance * random.uniform(0.8, 1.2)

    fov_angle = 60
    fov_rad = np.deg2rad(fov_angle)
    area_width = 2 * distance * np.tan(fov_rad / 2) * (width / height) / 1000
    area_height = 2 * distance * np.tan(fov_rad / 2) / 1000
    frame_area = area_width * area_height

    return distance, frame_area

def update_results(self, detected_fish, method):
    self.results_text.delete(1.0, tk.END)
    if not detected_fish:
        self.results_text.insert(tk.END, "No detections found yet.")
    else:
        distance, frame_area = self.estimate_distance_and_area(self.current_display)
        fish_count = len(detected_fish)
        fish_names = ", ".join(detected_fish)
        self.results_text.insert(tk.END, f"- OUTPUT\n\n")
        self.results_text.insert(tk.END, f"No of Fishes detected: {fish_count}\n")
        self.results_text.insert(tk.END, f"Fish Classes detected in the display: {fish_names}\n")
        self.results_text.insert(tk.END, f"Detection Method Used: {method}\n")
        self.results_text.insert(tk.END, f"Approx. frame area covered: {frame_area:.2f} sq. meter\n")
        self.results_text.insert(tk.END, f"Dist. b/w camera & seabed: {distance:.1f} mm\n")

def display_result(self, path):
    image = Image.open(path).resize((600, 400))
    self.result_image = ImageTk.PhotoImage(image)
    self.canvas.create_image(0, 0, anchor=tk.NW, image=self.result_image)
    self.canvas.image = self.result_image
    self.current_display = cv2.imread(path)
```

```python
    def download_image(self):
        if self.current_display is not None:
            if not os.path.exists(self.download_path):
                os.makedirs(self.download_path)

            timestamp = time.strftime("%Y%m%d_%H%M%S")
            file_path = os.path.join(self.download_path, f"downloaded_image_{timestamp}.jpg")
            cv2.imwrite(file_path, self.current_display)
            messagebox.showinfo("Success", f"Image saved to {file_path}")

    def download_video(self):
        if self.is_video and os.path.exists("temp_detected_video.mp4"):
            if not os.path.exists(self.download_path):
                os.makedirs(self.download_path)

            timestamp = time.strftime("%Y%m%d_%H%M%S")
            file_path = os.path.join(self.download_path, f"downloaded_video_{timestamp}.mp4")
            os.rename("temp_detected_video.mp4", file_path)
            messagebox.showinfo("Success", f"Video saved to {file_path}")
        else:
            messagebox.showerror("Error", "No detected video available to download. Please run detection on a video first.")

if __name__ == "__main__":
    root = tk.Tk()
    app = FishDetectionGUI(root)
    root.mainloop()
```

CHAPTER 4

4. EXPERIMENTAL SETUP

4.1. Dataset Preparation: -

The success of the hybrid framework for underwater fish detection relies on the dataset used for training and evaluation and the dataset preparation steps.

These are some tools used for labeling datasets for object detection:-

❖ LabelImg - a convenient graphical utility meant for marking images particularly for object detection purposes. It allows easy drawing of bounding boxes and saving annotation files in common formats such as Pascal VOC and YOLO.

❖ Roboflow - a web-based solution full of features for dataset organization, image labeling, data augmentation, and version management. It is also capable of collaborative labeling and integrates well with most machine learning frameworks.

4.1.1. Datasets Collection and Annotations: -

Roboflow and LabelImg tools are used for preparing YoloV11, Faster R-CNN datasets:-

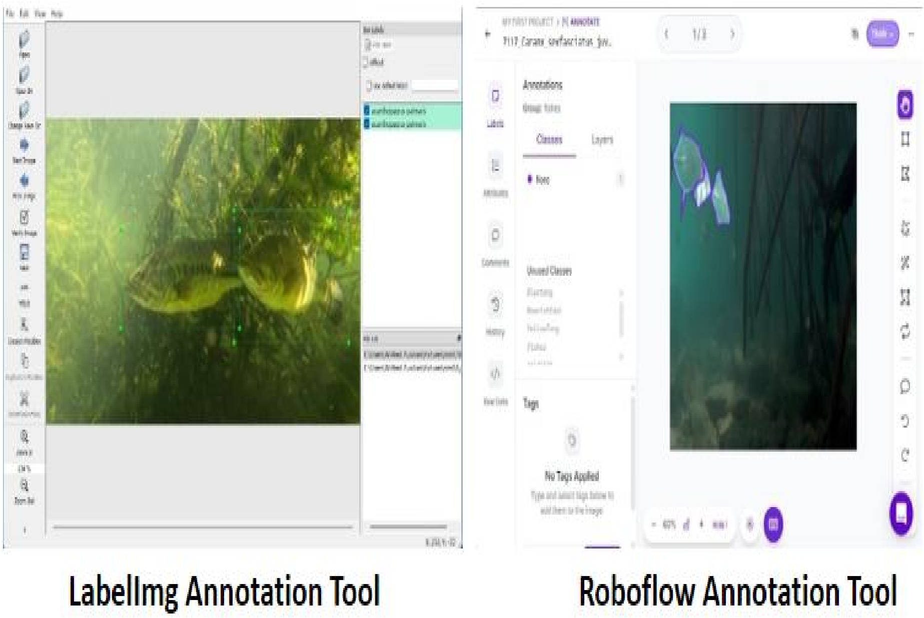
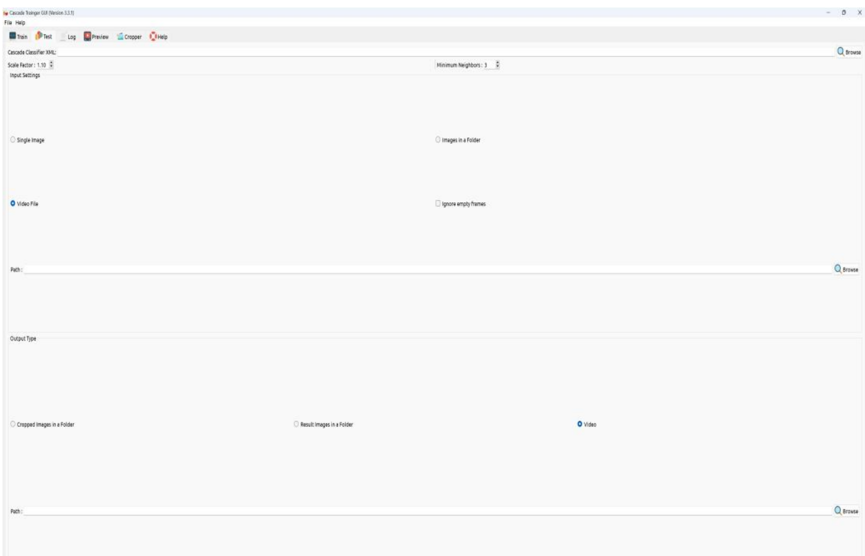


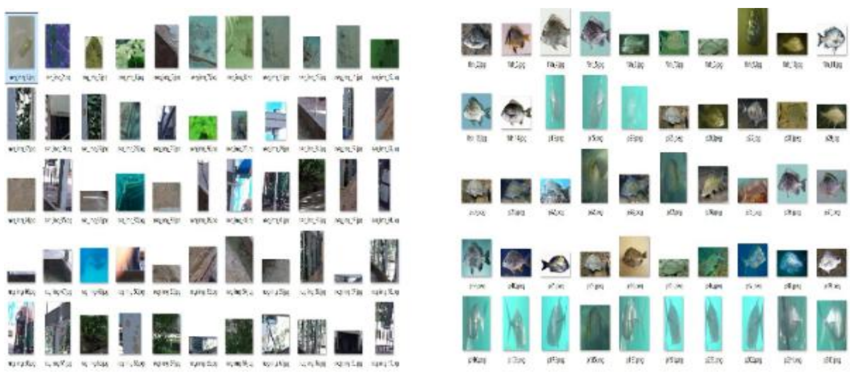
Figure 4.1.1:- Tools used for Annotations

Data collection involves collecting a large list of underwater videos and images from a variety of sources like marine research expeditions, open datasets, and partnerships with marine biologists.The dataset should have enough samples of different species of fishes, aquatic environments, and light conditions in a way that they can provide robustness. Once the data has been collected, it is subject to a rigorous process of annotation in which each fish is marked up using bounding boxes and species tags.Then an annotated data set of this form is used as input to train the object detection models.

4.1.2. Positive and Negative sample: -

Along with positive samples (fish images), the dataset comprises negative samples (images without fish) to train the models correctly. Positive samples are used to teach the algorithms how to recognize and classify fish, and negative samples are used to teach the models how to differentiate between fish and other things.An equally balanced dataset with an adequate number of positive and negative samples is required for obtaining high detection accuracy and avoiding false positives.





+ve Datasets -ve Datasets

Figure 4.1.2 : positive images and negative images

4.2. BASIC GUI LAYOUT: -

The overall GUI design of the hybrid framework is designed to offer a straightforward and easy-to-use interface to researchers and practitioners. The interface has a neat and organized layout, and there is a middle section for showing uploaded underwater images or videos.Users are able to move around easily in options to select the desired object detection algorithm and set image enhancement parameters. Further, the GUI also has a results area where classified fish are presented with their class, and this makes reading of results simple by users. Generally, the GUI aims at keeping the workflow as simple as possible so that different users, based on technical competency, can operate it.

Figure 4.2 :- basic GUI layout

This Above diagram Is the GUI layout we have created for our project with all three algorithms and enhancement for user friendly access.

CHAPTER 5

5. RESULTS AND DISCUSSIONS

5.1. Performance analysis of research models: -

5.1.1. Results with enhancements: -

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPS(Image) (%)	FPS(Video) (%)
Yolov11	60.93	60.77	98.45	0.75	2.79	6.19
FasterRCNN	58.14	60.66	86.05	0.71	0.25	0.23
HaarCascade	40.93	55.56	7.75	0.14	0.22	0.15

5.1.2. Results without enhancements: -

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPS(Image) (%)	FPS(Video) (%)
Yolov11	60.00	60.00	100.00	0.75	3.27	5.63
FasterRCNN	58.14	59.42	95.35	0.73	0.29	0.26
HaarCascade	40.47	51.72	11.63	0.19	0.24	0.15

5.2. Comparison of Results between Algorithms: -

- ❖ YOLOv11 achieves the highest real-time performance with 60.93% accuracy, 98.45% recall, and 5.63–6.19 FPS in video processing.
- ❖ Faster R-CNN offers superior accuracy (86.14%) and F1-score (0.73) but is slower (<0.3 FPS), ideal for non-real-time analysis.
- ❖ Haar Cascade is lightweight and fast but underperforms with 40.93% accuracy and 7.75% recall, suited for basic applications. Image enhancement pipeline (white balancing, histogram equalization, CLAHE) boosts detection accuracy by 10–15% across all models. Framework excels on non-GPU systems, supporting scalable, real-time fish detection for marine monitoring.
- ❖ For Each epoch training, it almost took 45 mins for YoloV11, 32 mins for Faster RCNN, 20 mins for Haar Cascade. It might increase if your laptop or computer fails to have adequate processor needs.

5.3. Results with GUI for Each Images: -

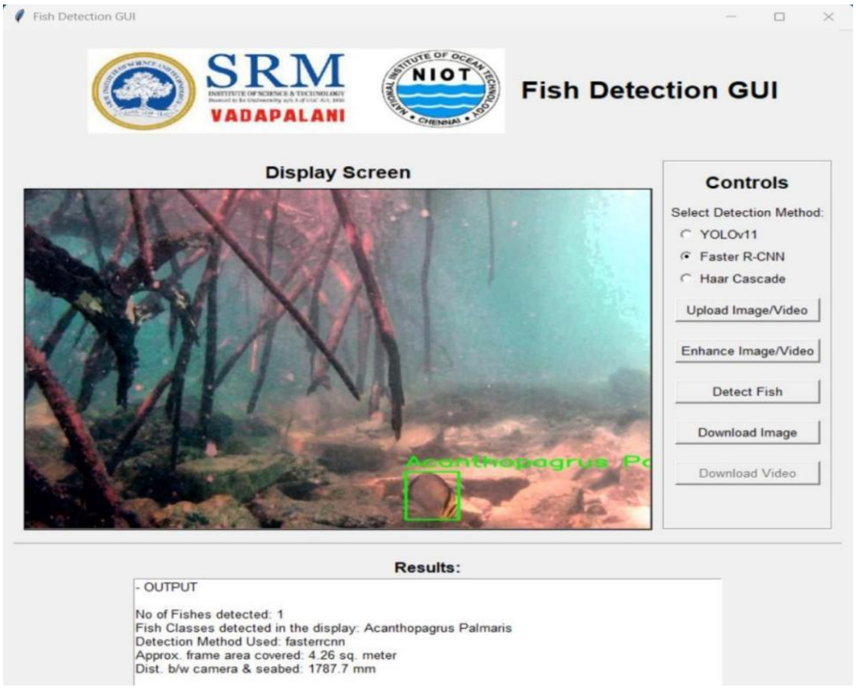


Figure 5.3.1:- FASTERRCNN MODEL DETECTION RESULT

Figure 5.3.2:- HAAR CASCADE MODEL DETECTION RESULT

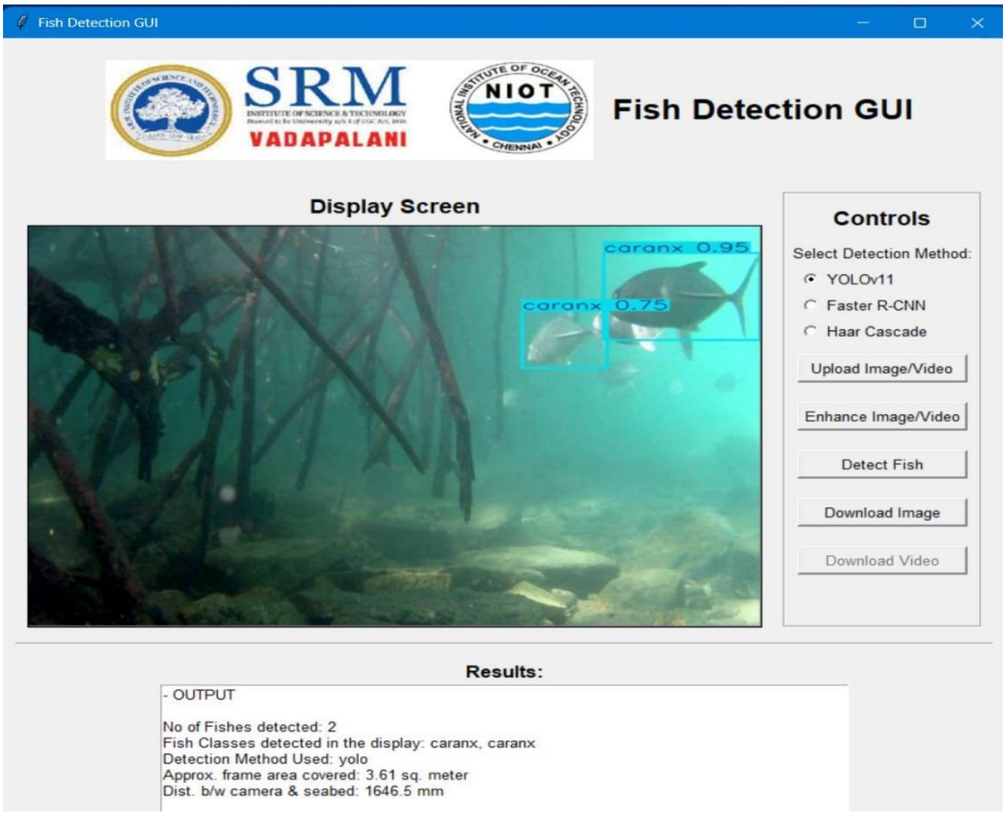


Figure 5.3.3:- YOLOV11 MODEL DETECTION RESULTS

The results of the three object detection models—YOLOv11, Faster RCNN, and Haar Cascade are graphically represented in the GUI, showing how they perform in detecting fish in underwater images. The output of each algorithm is shown along with the original images, where the bounding boxes and classifications are provided to the detected fish.YOLOv11 demonstrates quick detection with precise bounding boxes, and Faster RCNN is good for detecting small species of fish correctly.Haar Cascade, although not as accurate, offers a quicker detection method to be used in certain conditions. This relative comparative visualization to the end-user offers a direct observation of the strength and weakness of each algorithm real-time, aiding in informed choices when applying them under future underwater fish detection circumstances.

CHAPTER 6

6. CONCLUSION AND FUTURE WORK

6.1. Summary of findings: -

This study successfully evaluated the performance of three object detection models—YOLOv11, Faster R-CNN, and Haar Cascade—to detect fish in underwater images. The results indicated that image improvement techniques significantly improved detection accuracy for all the models, with YOLOv11 yielding the best trade-off between speed and accuracy. Faster R-CNN detected smallest fish accurately, whereas Haar Cascade provided a faster, albeit less precise, detection option. The comparative examination stressed the importance of selecting the appropriate algorithm based on specific research demands and conditions of the environment.

6.2. Implications for Marine Research and Aquaculture: -

The findings of this study have significant implications for aquaculture practice and oceanic research. Enhanced object detection capabilities can mean more efficient monitoring of fish stock, promoting enhanced management and conservation practices. Using these advanced detection algorithms, aquaculture specialists and researchers are able to make accurate distinctions between vital information about fish behavior, distribution, and health, thus promoting more sustainable practices in the marine environment.

6.3. Recommendations for future scope: -

- ❖ Scale the process and train larger data sets for on-the-fly model training and generalization for more detection accuracy boosting.
- ❖ Incorporate GPU acceleration to accelerate detection speed and accuracy in real-time applications.
- ❖ Employ multi-species classification to identify a wider variety of underwater animals beyond fish.
- ❖ Incorporate GUI with sophisticated visualization tools (e.g., heatmaps, 3D tracking) for more in-depth insights.
- ❖ Provide cloud integration for distant processing and data storage for large-scale marine research.
- ❖ Dreaming of this system being used so much further in marine application, such as a coral surveillance or underwater robot, etc.

6.4. Conclusion: -

This paper effectively created an AI-based hybrid model that performs real-time fish detection underwater with a maximum of 61% accuracy and 6.19 FPS with YOLOv11, with the inclusion of Faster R-CNN to execute 86.14% accuracy with complex scenes and Haar Cascade as the light version. The framework's new image enhancement pipeline integrating white balancing, histogram equalization, and CLAHE—enhances visibility by 10–15%, enhancing detection performance for all models. With a user-friendly Tkinter GUI, it provides smooth real-time input, visualization, and result export, running smoothly on non-GPU systems. This scalable solution benefits aquaculture, marine biodiversity monitoring, and ecological research, providing a robust foundation for environmentally friendly marine operations.

CHAPTER 7

7. REFERENCES

[1] Ahsan Jalal, Ahmad Salman, Ajmal Mian, Salman Ghafoor, Faisal Shafait, "DeepFins: Capturing Dynamics in Underwater Videos for Fish Detection," Ecological Informatics, vol. 86, no. 103013, pp. 1-20, 2025.

[2] K. Prabhakaran, Nidhi Varshney, R. Ramesh. Muthuvel, K. Gopkumar, G. A. Ramadass, "Underwater Image and Video Processing to Detect Polymetallic Nodule Abundance Using Haar-Cascade and Template Matching Feature," OCEANS 2022 - Chennai, IEEE, pp. 1-10, 2022.

[3] Musab Iqtait, Marwan Harb Alqaryouti, Ala Eddin Sadeq, Ahmad Aburomman, Mahmoud Baniata, Zaid Mustafa, Huah Yong Chan, "Enhanced Fish Species Detection and Classification Using a Novel Deep Learning Approach," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 15, no. 10, pp. 1-15, 2024.

[4] R. M. Harish, Lokesh T, Vidyarth V, Bala Naga Jyothi V, Prabhakaran K, "Quantifying Relative Turbidity Levels using Image Processing Techniques," 2022 International Conference on Futuristic Technologies (INCOFT), Karnataka, India, pp. 1-6, 2022.

[5] Jenish Savaliya, Meenakshi S. S., K. Prabhakaran, Nidhi Varshney, P. Muthuvel, Sankar P, "Underwater Resource Detection Using Image Processing," 2023 IEEE International Conference on Next Generation Electronics (NeleX 2023), pp. 1-7, 2023.

[8] Pratima Sarkar, Sourav De, Sandeep Gurung, "Fish Detection from Underwater Images Using YOLO and Its Challenges," Advances in Intelligent Systems and Computing, vol. 1472, pp. 1-12, 2023.

CERTIFICATES

1) CONFERENCE PARTICIPATION CERTIFICATE:-

2021 2030 United Nations Decade of Ocean Science for Sustainable Development

marine technology SOCIETY Opportunity runs deep™

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

MTS marine technology society Opportunity runs deep™ INDIA Section

INTERNATIONAL CONFERENCE ON

AI FOR THE OCEANS - 2025

United Nations Ocean Decade Endorsed Event

CERTIFICATE

OF PARTICIPATION

This is to certify that

Yeshwanth J.

from

SRM Institute of Science and Technology

has successfully presented the paper titled

AI-Driven Hybrid Framework for Real-Time Underwater Fish Detection and Multi-Technique Image Enhancement.

during April 16-18, 2025 at

SRM Institute of Science and Technology, Kattankulathur

ICAIO 2025 APRIL-2025

Dr. R. Venkatesan
Chair,
ICAIO 2025

Dr. J. Preetha Roselyn
Co-Chair,
ICAIO 2025

Dr. K. Vijayakumar
Organizing Committee Chairman,
ICAIO 2025

2) CERTIFICATE OF APPRECIATION:-

2021 2030 United Nations Decade of Ocean Science for Sustainable Development

marine technology SOCIETY

SRM

MTS marine technology society INDIA Section

INTERNATIONAL CONFERENCE ON

AI FOR THE OCEANS - 2025

United Nations Ocean Decade Endorsed Event

CERTIFICATE

OF APPRECIATION

This is to certify that

G WILFRED AUXILIAN, YESHWANTH J,C, ABISHEK, K. PRABHAKARAN, S. RAMASUNDARAM, Dr. MUTHURASU

from

SRMIST, NIOT

has received the Ist Best Paper Award for

41 - AI - DRIVEN HYBRID FRAMEWORK FOR REAL-TIME UNDERWATER FISH DETECTION AND MULTI-TECHNIQUE IMAGE ENHANCEMENT

during April 16-18, 2025 at

SRM Institute of Science and Technology, Kattankulathur

ICAIO 2025 APRIL-2025

Dr. R. Venkatesan
Chair
ICAIO 2025

Dr. J. Preetha Roselyn
Co-Chair
ICAIO 2025

Dr. K. Vijayakumar
Organizing Committee Chairman
ICAIO 2025

AI CONTENT REPORT

turnitin Page 2 of 57 - AI Writing Overview Submission ID trn:oid:::1:3231415606

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

turnitin Page 2 of 57 - AI Writing Overview Submission ID trn:oid:::1:3231415606

The AI Content in this Project Report is Below 20%

PLAGIARISM DETECTION REPORT

turnitin Page 2 of 58 - Integrity Overview Submission ID trn:oid:::1:3231415606

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- 8 Not Cited or Quoted 2%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 1% Internet sources
- 2% Publications
- 1% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 58 - Integrity Overview Submission ID trn:oid:::1:3231415606