National College of Ireland

Configuration Manual

MSc Research Project
MSc in Artificial Intelligence

Kshitija Chavan
Student ID: x22123598

School of Computing
National College of Ireland

Supervisor: Abdul Razzaq

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kshitija Chavan
Student ID:	x22123598
Programme:	MSc in Artificial Intelligence **Year:**2023
Module:	Research Project
Lecturer:	Abdul Razzaq
Submission Due Date:	31st January, 2024
Project Title:	AI-driven Autonomous Vehicle using Yolov8 and Deep learning
Word Count:	872 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Kshitija Chavan

Date: 31st January, 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kshitija Chavan
Student ID: x22123598

1 Introduction

The manual provides comprehensive details on setting up the software and outlines the necessary hardware requirements for constructing the entire system based on the models. It serves as a practical guide, offering a replicable overview of the research work. Additionally, the manual includes Python code evaluations, seamlessly integrating them with the interface for a cohesive understanding of the implemented processes.

2 System Specification

Operating System: Windows 11
Intel i7 CPU with 2.50 GHz
64 bit operating system
RAM: 16 GB
GPU: Nvidia Geforce RTX 3050

3 Software Specifications

In this section, we examine the software requirements that were used in implementing the model. The platform of choice is Visual Studio Code, and the major programming language is Python. Moreover, a number of libraries and packages have been included to ensure accurate and organized outcomes.

- Python 3.8.10
- Pandas
- Tkinter
- NumPy
- Matplotlib
- Sklearn
- OpenCV
- TensorFlow
- Yolov8
- Dlib
- PyGame
- PyAudio

4 Implementation and Evaluation

```
def laneDetector(image, display_result):
    image_shape = image.shape
    grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    kernel_size = 9
    smoothed_image = cv2.GaussianBlur(grayscale_image, (kernel_size, kernel_size), 0)
    min_val = 60
    max_val = 150
    edges_image = cv2.Canny(smoothed_image, min_val, max_val)
```

laneDetector Function: This function uses OpenCV to analyze an input picture and find lanes. A mask is made to focus on the area of interest, the picture is converted to grayscale, Gaussian blur and Canny edge detection are used, and Hough lines detection is used to identify lane lines. After that, the identified lanes are blended and projected into the original picture.

```
def estimate_vehicle_distance(input_image, yolo_detector):
    result_image = yolo_detector.detect_image(Image.fromarray(input_image))
    # cv2.imshow('Vehicle Distance Estimator', cv2.resize(np.asarray(result_image), (600, 400), i
    return np.asarray(result_image)


def preprocess_image_for_keras(input_img):
    target_image_size_x = 100
    target_image_size_y = 100
    processed_img = cv2.resize(input_img, (target_image_size_x, target_image_size_y))
    processed_img = np.array(processed_img, dtype=np.float32)
    processed_img = np.reshape(processed_img, (-1, target_image_size_x, target_image_size_y, 1))
    return processed_img


def detect_video(video_path, yolo_detector):
    # Load the steering wheel image
    steering_wheel = cv2.imread('resources/steering_wheel_image.jpg', 0)
    rows, cols = steering_wheel.shape

    smoothed_angle = 0
    frame_counter = 0
    frame_rate = 10
```

estimate_vehicle_distance Function: To determine how far away the cars are in the supplied image, this function uses YOLO (You Only Look Once) object recognition.
With this function, a picture can be resized and reshaped in order to get it ready for input into a Keras model.
detect_video Function: This function examines each frame of a video file. It rotates an image of the steering wheel in accordance with the predicted steering angles using the autopilot model, smoothes the steering angle, calculates vehicle distances using YOLO, and carries out lane detection. An image of the rotating steering wheel is included in the final result that is shown.
detect_image Function: Analogous to function detect_video, however it handles a single image rather than a video.

```
import cv2


face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Cascade Classifier Initialization:

- It loads the Haar Cascade classifier for face detection using **cv2.CascadeClassifier**.
- The classifier file, 'haarcascade_frontalface_default.xml', is assumed to be present in the same directory as the script.

```
def sendAlertEmail(person_name=Constants.person,  family_member_email_ids=[Constants.email]):
    try:
        # Get the current timestamp
        current_time = datetime.datetime.now()

        # Format the timestamp
        formatted_time = current_time.strftime("%Y-%m-%d %H:%M:%S")

        # Include the formatted timestamp in the message
        message = 'Hello, \nContact {} as soon as possible because {} is found in a drowsy situation while driving. This message was sent

        # Create a yagmail SMTP connection
        yag = yagmail.SMTP("autoemailsender2@gmail.com", "tczewxnxfrpviped")

        # Send the email to family members
        yag.send(to=family_member_email_ids, subject='Drowsiness Alert Message', contents=message)

        # Close the yagmail SMTP connection
        yag.close()

        print("Email sent successfully")
    except Exception as e:
        print('Email not sent due to:', e)
```

Function sendAlertEmail:

The function takes two parameters: person_name (defaulting to Constants.person) and family_member_email_ids (defaulting to [Constants.email]).

Inside the function:
It gets the current timestamp using datetime.datetime.now() and formats it.
Constructs an email message containing information about a detected drowsy situation while driving.
Uses the yagmail library to create an SMTP connection with credentials (autoemailsender2@gmail.com and password "tczewxnxfrpviped").
Sends the email to the specified family members with the subject "Drowsiness Alert Message" and the constructed message.
Closes the yagmail SMTP connection.
Prints a success message if the email is sent; otherwise, it prints an error message.

```python
# Load face cascade for drawing rectangles around detected faces
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Define a function to calculate and return eye aspect ratio
def calculate_eye_aspect_ratio(eye_landmarks):
    A = distance.euclidean(eye_landmarks[1], eye_landmarks[5])
    B = distance.euclidean(eye_landmarks[2], eye_landmarks[4])
    C = distance.euclidean(eye_landmarks[0], eye_landmarks[3])

    ear = (A + B) / (2 * C)
    return ear

# Load face detector and predictor, using a dlib shape predictor file
face_detector = dlib.get_frontal_face_detector()
shape_predictor = dlib.shape_predictor("dataset\shape_predictor_68_face_landmarks.dat")

# Extract indexes of facial landmarks for the left and right eye
(left_eye_start, left_eye_end) = face_utils.FACIAL_LANDMARKS_IDXS['left_eye']
(right_eye_start, right_eye_end) = face_utils.FACIAL_LANDMARKS_IDXS['right_eye']

# Start webcam video capture
video_capture = cv2.VideoCapture(Constants.source)

while True:
    # Read each frame, flip it, and convert to grayscale
    ret, current_frame = video_capture.read()
    cv2.imshow('Video Feed', current_frame)

    current_frame = cv2.flip(current_frame, 1)
    grayscale_frame = cv2.cvtColor(current_frame, cv2.COLOR_BGR2GRAY)

    # Detect facial points using the detector function
    detected_faces = face_detector(grayscale_frame, 0)
```

Initialize Pygame:
Pygame is initialized to handle sound, and an alert sound is loaded.

Initialize Variables:
Various variables are initialized, including thresholds, frame counters, and paths.

Load Face Cascade and Shape Predictor:
Haarcascades for face detection and a shape predictor for facial landmarks are loaded.

Define a Function to Calculate Eye Aspect Ratio (EAR):
The function calculate_eye_aspect_ratio computes the EAR using the Euclidean distances between facial landmarks.

Start Webcam Capture:

The code starts capturing video from the webcam.

5 Results

```
284/284 [==============================] - 1s 3ms/step
Precision: 0.763170188220263
Recall: 0.8735961242017177
F1 Score: 0.8146581628369097
Accuracy: 0.8735961242017177
```

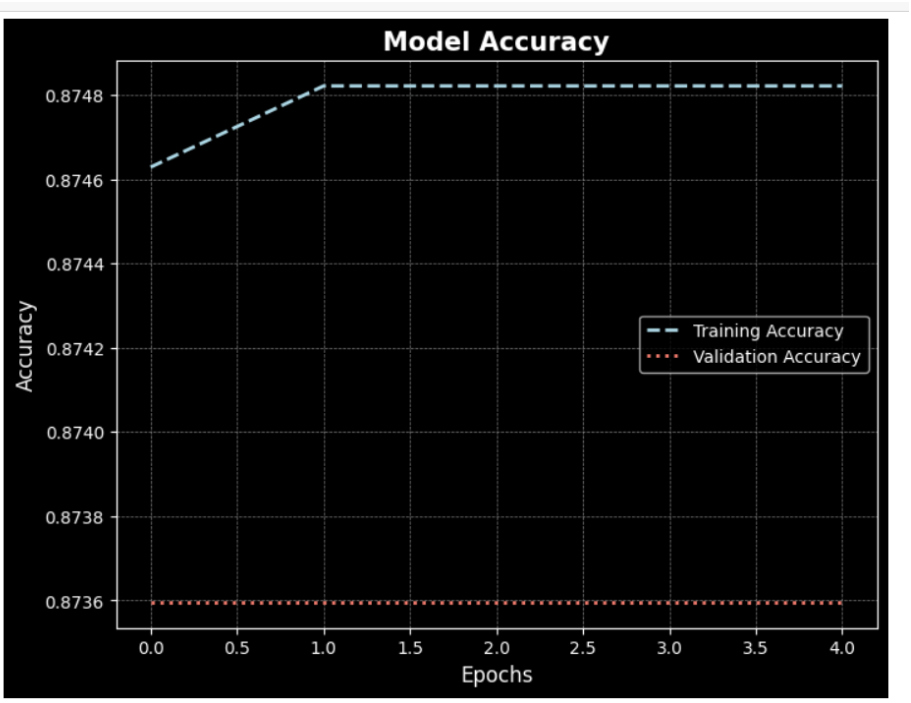


Figure 1. Model's Accuracy Vs Epochs graph

The neural network model's development during five training epochs is documented in the training log that is supplied. The model has a consistent training accuracy of approximately 87.48% across epochs, accompanied by a steady decline in validation loss, suggesting enhanced adaptability to novel data. The best-performing model is preserved by a checkpoint system indicated by the periodic storage of model weights to "Stearing.h5". The general pattern indicates that the model will converge as it adjusts its parameters, bringing it closer to the goal of reducing error and improving precision.

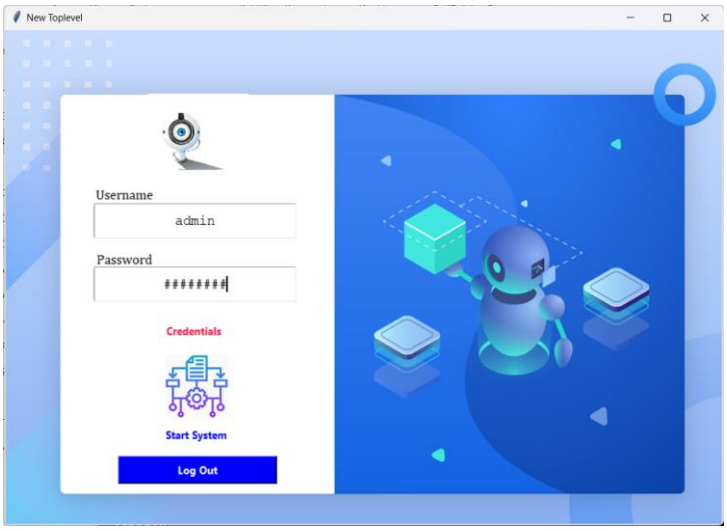


Figure 2. Login Page

The user interface (UI) highlights a particular set of features. Before getting access to the system, users must log in using credentials. A safe and authenticated user experience is guaranteed by this login process. After logging in successfully, users may submit a picture or a video using the user interface. An essential part of the project's functionality is the upload capability, which lets users enter visual data into the system for processing or analysis. By requiring a login, the UI's picture and video uploading features are made accessible only to authorized users, providing an extra degree of protection and control.

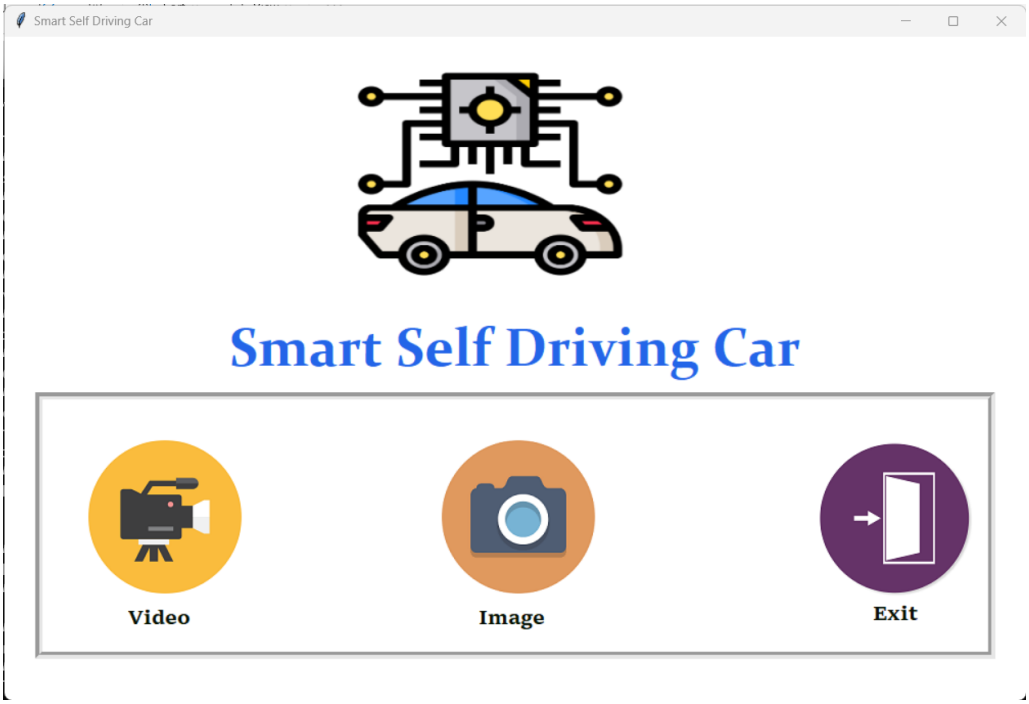
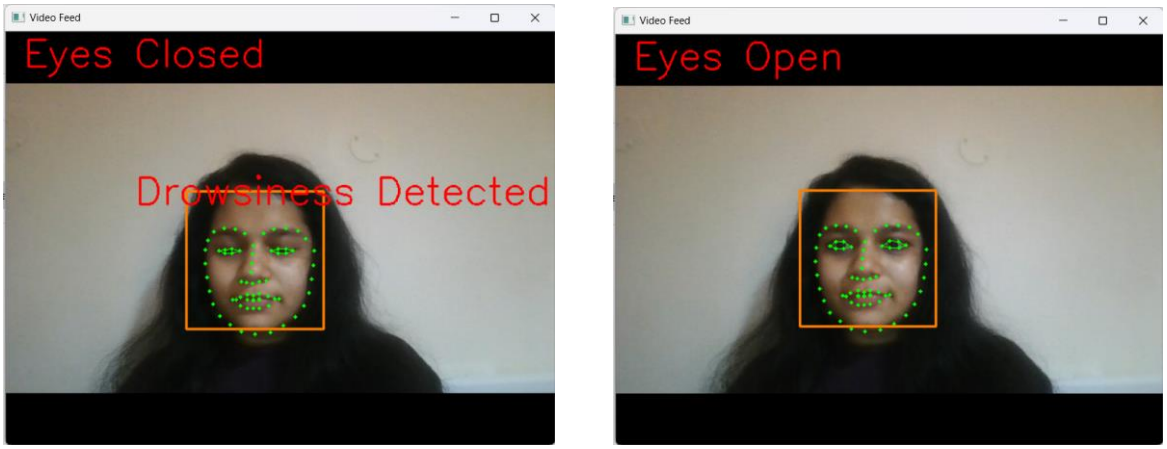


Figure 3. Dashboard



A. Eyes Open B. Eyes Closed

Figure 4. Drowsiness Detection

Figure 5. Lane and Vehicle Detection, Distance Estimation

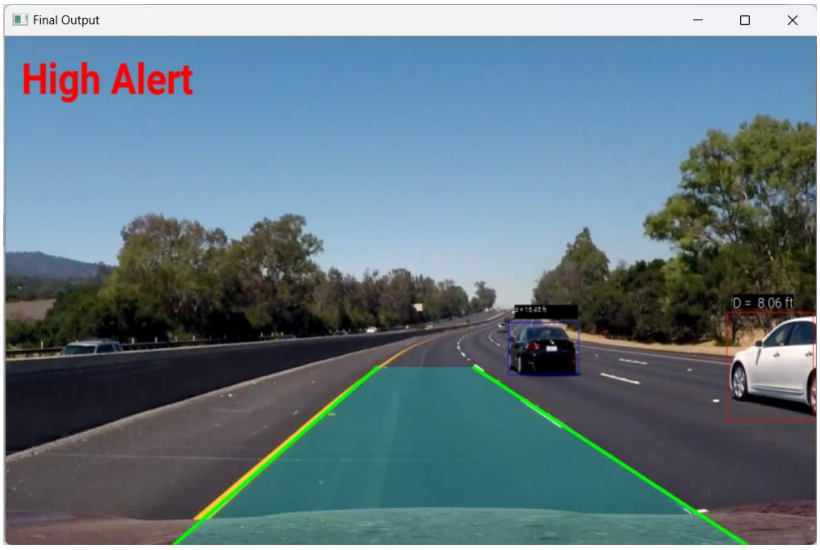


Figure 6. Alert System shown on Drowsiness detection

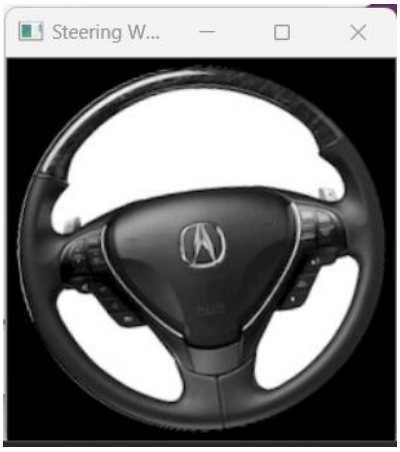


Figure 7. Steering Wheel Automation

References

M. Uma, S. Abirami, M. Ambika, M. Kavitha, S. Sureshkumar and K. R, "A Review on Augmented Reality and YOLO," *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2023, pp. 1025-1030, doi: 10.1109/ICOSEC58147.2023.10275842.

M. Alhafnawi et al., "A Survey of Indoor and Outdoor UAV-Based Target Tracking Systems: Current Status, Challenges, Technologies, and Future Directions," in *IEEE* Access, vol. 11, pp. 68324-68339, 2023, doi: 10.1109/ACCESS.2023.3292302.

A. B. Amjoud and M. Amrouch, "Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review," in *IEEE* Access, vol. 11, pp. 35479-35516, 2023, doi: 10.1109/ACCESS.2023.3266093.