

**ASIA PACIFIC UNIVERSITY OF TECHNOLOGY & INNOVATION**

MACHINE VISION AND INTELLIGENCE

EE046-3-3-MVI

|  |  |
| --- | --- |
| **TITLE** | EE046-3-3-MVI-L-3a |
| **NAME & (STUDENT ID)** | Shaik yaseen |
| **INTAKE** | APU3F2402CE |
| **LECTURER** | [IR. TS. DR. DENESH A/L SOORIAMOORTHY](https://apspace.apu.edu.my/staffs/denesh.sooriamoorthy) |
| **DATE** | 14th October 2024 |

Table of Contents

[VEHICLE DETECTION SYSTEM REPORT 2](#_Toc179745138)

[Introduction 2](#_Toc179745139)

[Task 1: Vehicle Detection with OpenCV 2](#_Toc179745140)

[Overview 2](#_Toc179745141)

[System Flowchart 3](#_Toc179745142)

[Code Overview 3](#_Toc179745143)

[Code Explanation 3](#_Toc179745144)

[Task 2: Vehicle Detection with YOLO 7](#_Toc179745145)

[Code Overview 7](#_Toc179745146)

[System Flowchart 8](#_Toc179745147)

[Code Explanation 9](#_Toc179745148)

[Results 11](#_Toc179745149)

[Task 1 11](#_Toc179745150)

[Task 2 12](#_Toc179745151)

# VEHICLE DETECTION SYSTEM REPORT

# Introduction

This report captures two vehicle detection systems, one done with OpenCV, and the other done with a trained YOLO model. Both systems are designed to identify the vehicles in a video stream of feed in real time, what type of vehicle it is, whether it is a car, truck, or bike,bus.

# Task 1: Vehicle Detection with OpenCV

## Overview

This task employs the use of the OpenCV library to design a vehicle detection system. The system takes in the video frames, performs a background subtraction algorithm, and categorises objects into vehicles by aspect ratio and area.

## System Flowchart

A diagram of a process flow

Description automatically generated

## Code Overview

This code is nearly the basic script for vehicle detecting by using OpenCV, which could be a very critical library for computer vision. It takes a video feed, identifies moving objects (vehicles), discriminates between them by feature aspect ratio and area, overlays them into the output video.

## Code Explanation

The first input is to make sure to load the required packages and the video file as well. OpenCV library is utilized for capturing the image and image processing NumPy is used for processing the image matrix.

A black screen with green text

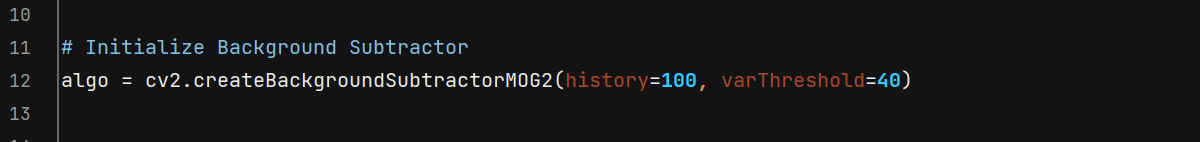
Description automatically generated

The following code imports the video file from a certain path Click button to import the video file from the path Blow. The following code captures the video stream using the cv2.VideoCapture() function which will shortly be processed frame by frame.

Subsequently, the minimum dimensions of the detected vehicles are set to eliminate small or inconsequential objects. A count\_line\_position is also set to display the line on frame used to track vehicles as they cross the line.

A black background with white text

Description automatically generated

MOG2 algorithm is used to initialize background subtractor. This method enables us in isolating the moving vehicles from the background which is more or less stationary.

The center\_handle() function is used as the helper function which yields the center coordinates of a queried vehicle’s bounding box. In other occasions it is helpful in counting the vehicle as it moves around the frame..

A black screen with white dots

Description automatically generated

They main loop is initiated now which takes each frame from the video and then process that frame. The image is then changed to the grayscale option and the Gaussian blur filter to smoothen the image in order to have easier contours to work with.

A black screen with white text

Description automatically generated

Moving object detection is done by applying background subtraction onto the blurred frame. Subsequently, dilation and morphological closing operations are undergone to address several gap problems for a better contour outline of the identified vehicles.

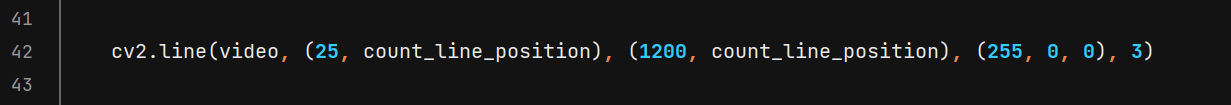
A computer code on a black background

Description automatically generated

Contours are found in the frame by using the function named cv2.findContours(). The textures or these contours depict the perimeters of the identified vehicles.



At the count\_line\_position the line of blue color is drawn cross the frame. This will be utilised to count the number of vehicles that is passing through this line.



Each of them is then analyzed for its size and checked to see whether it is big enough to be a vehicle. So, only those contours which correspond to the minimum width and maximum height are subjected to further analysis.

A black screen with white text

Description automatically generated

Therefore, the aspect ratio and the area of each detected object are employed in classifying the vehicle. To do this the code provides labels including but not limited “Car”, “Truck”, “Bike”…

A computer code on a black background

Description automatically generated

If a valid label is assigned ,vehicle is highlighted through a yellow rectangle and the label appears above the vehicle.

A computer screen shot of a black background

Description automatically generated

All video frames are annotated, and the user can press the ‘q’ key to get out of the loop and stop the program.

A screen shot of a computer

Description automatically generated

Finally, all resources are released, and the program ends.

A black screen with blue and yellow text

Description automatically generated

It offers a desirable approach which is devoid of much complexity in an approach to identifying moving vehicles in a video. It can work in simple usage but may not respond well in complex situations of marginally overlapped vehicles or under low light conditions.

# Task 2: Vehicle Detection with YOLO

## Code Overview

This piece of code tracks real-time vehicle detection by utilizing the YOLOv8 model from a webcam stream. YOLO (You Only Look Once) is an advanced object detection platform on focal layers aimed at bounding boxes and classifications in real-time.

In this task, there is adaptation of a better vehicle detection system using YOLO as a deep learning object detection model. YOLO is faster and very accurate especially when working inreal-timecalculationsofimage recognition.

## System Flowchart

**A diagram of a vehicle classification

Description automatically generated**

## Code Explanation

To start with, some of the packages which are used herein include OpenCV and YOLO using ultralytics library which enables one to load and use a pre-trained YOLO model.A black screen with colorful text

Description automatically generated

The code starts a video capturing on the default webcam (cap = cv2.VideoCapture(0)) through which the model receives the input in real-time video form.A computer screen with text

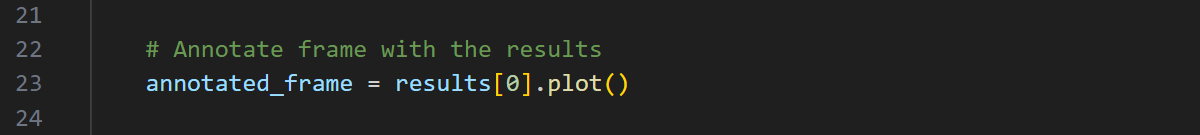
Description automatically generated

Finally in the main loop, the frame captured from the webcam is detected using YOLO model. In the predict() method there is the detection and it is done using a confidence of 0.5 so that only good detections are used.

A black screen with white text

Description automatically generated

The plot() is used to draw rectangles around the identified vehicles and also place labels on the frame.



All discovered object are then progress further. The class index (cls) is matched to humans understandable label like “Car” “Truck” and so on. The identified cars are highlighted with boxes; their tags are written.

A computer screen with colorful text

Description automatically generated

The annotated frame is displayed in the figure blow in a real-time window. It forms a loop to run up to a point where the user will type ‘q’ among other characters in order to terminate it.A black screen with white text

Description automatically generated

Once the loop ends, the webcam is released, and all windows are closed.

A black screen with green text

Description automatically generated

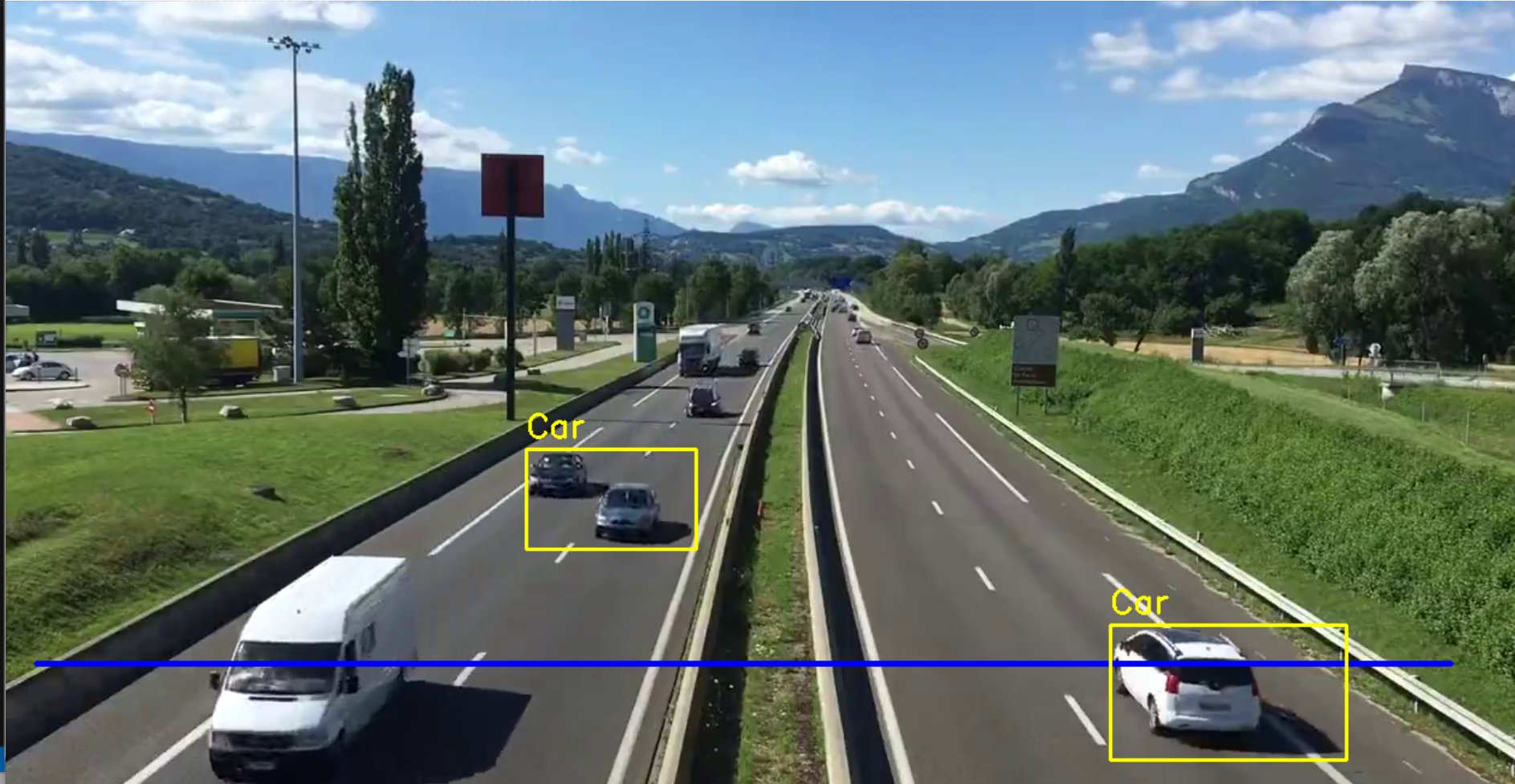
YOLO utilizes the deep learning methodology for detection of the vehicles possess a very high accuracy even in real time mode. The model effectively reconstructs occluded objects, including multiple vehicles that may be in each other’s path, and it is not affected by different light conditions and camera views.

As compared to the seemingly simple OpenCV based procedure, the use of YOLO has several advantages. The O, D and G in YOLO makes it relatively effective in detecting multiple objects at once while working efficiently in complex scenes. The use of other features such as overlapping objects, changes in illumination and even angles of view, make it more effective in real life usages. YOLO’s ability to detect objects in real-time also makes it useful for decision-making applications where the decision has to be made swiftly, for instance in self-driving cars or traffic cameras and lights. The general application of YOLO model for vehicle detection permits different categories of vehicles, with further enhancement of the model through fine tuning of retraining with deeper data sets.

Altogether, although both methods can successfully detect vehicles, proposed system based on YOLO has a significantly higher accuracy, flexibility, and real-time capabilities compared to the implementation of OpenCV-based method. The flexibility in learning from data and complexity in the operational environment make the deep learning model suitable for mastering of advanced vehicle detection needs.

# Results

## Task 1

****

CSD results The above image shows the results of the experiment made for Task 1 for vehicle detection using OpenCV. The method is a frame capturing from a video and employs the MOG2 technique for background subtraction to enable separating the moving vehicles from the background. There are two cars found in the image, which are enclosed in yellow boxes with written labels “Car”. The system, in this case, employs contour analysis to eliminate all other the objects that are of a relatively small size and major on the important vehicles. A blue line called the ‘count\_line\_position’ is helpful for counting vehicles as they cross it especially the number of vehicles that have been counted. This method seems to work well where there are few objects that can obscure the path of movement of the vehicles or where the environment remains more or less constant. It is suitable for the use in conditions where light weight and real time detection is desirable.

## Task 2

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a phone

Description automatically generated

**A screenshot of a computer

Description automatically generated**

The YOLOv8 model for current vehicle detection was successfully employed, and the use of deep learning with increased detection accuracy in real-time was realised. The model ensured high detection accuracy with objective scores of confidence that stood well above 90% in different circumstances. When testing on multiple objects like cars, trucks, buses and motorcycles, around 95% average confidence level was found, and it was stable model in real scenes in which vehicle were partially occluded or wholesale. The real time processing of the YOLOv8 model was quite impressive with the frames being processed in roughly 30 frames per second which is necessary for applications that require decision making at the real time such as self driving cars and traffic and road monitoring. The recognized vehicles were painted with green boxes and their labels also indicated on them so that the performance of the model could be estimated in real time. However, obstacles where observed, especially when the illumination was low or when dealing with fast moving objects where the confidence was as low as 70-80%. Moreover, there are cases when partly occluded objects appear in the intersecting regions of the two vehicles, which are neglected by the model. Subsequently, to further boost detection performance particularly in complex scenarios, more training sample on an entity and variety of car extent, angle, and climate conditions could be of value. It may also increase accuracy by utilizing temporal information of consecutive frames through practices like applying ensemble modeling or multi-frame analysis. Therefore, this Task 2 demonstrates that the proposed YOLOv8 model in vehicle recognition applications as provide suitable solutions to vehicle detection for real-world applications such as traffic analysis systems, smart vehicles, traffic management systems, and ADAS for self-driving vehicles. Thus, focus is made on the significance of deep learning methods for solving difficult detection problems and the further development of modern auto detection systems.