Traffic Monitoring System Using Python-OpenCV & YOLOv8

A PROJECT REPORT

Submitted by

Meenakshi Katroth(B192491)
Anish Kumar Cindi(B191167)
Uday Kumar Naik Mudavath (B192476)

Of

Bachelor of Technology

Under the guidance of

Mr.Ranjith Garnepudi

Asst. Prof, CSE, RGUKT BASAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

BASAR, NIRMAL(DIST)

TELANGANA - 504107

Traffic Monitoring System Using Python-OpenCV & YOLOv8

Project Report submitted to
Rajiv Gandhi University of Knowledge Technologies, Basar
for the partial fulfillment of the requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

by

Meenakshi Katroth(B192491)
Anish Kumar Cindi(B191167)
Uday Kumar Naik Mudavath (B192476)

Under the guidance of

Mr.Ranjith Garnepudi
Assistant Professor,
CSE, RGUKT BASAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES BASAR
MAY 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

CERTIFICATE

This is to certify that the Project Report entitled 'Traffic Monitoring System Using Python-OpenCV&YOLOv8'submitted **by K.Meenakshi(B192491), C.Anish Kumar(b191167), M.Uday Kumar Naik (B192476)** Department of Computer Science and Engineering ,Rajiv Gandhi University of Knowledge & Technologies , Basar; for partial fulfillment of the requirement of the degree of Bachelor Technology in Computer Science & Engineering is a Bonafide record of the work & Investigations carried out by them under my supervision & guidance.

PROJECT SUPERVISOR	HEAD OF DEPARTMENT
Mr. Ranjith Garnepudi	**Mr. Dr B.Venkat Raman**
Assistant Professor.	**Assistant Professor.**

EXTERNAL EXAMINER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

DECLARATION

We hereby declare that the work presented in this project report titled "**Traffic Monitoring System Using Python-OpenCV & YOLOv8**", submitted to **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR**, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**, is an authentic record of our original work carried out under the supervision of **Mr. Ranjith Garnepudi,** Assistant Professor, Department of Computer Science and Engineering, RGUKT Basar.

We further declare that the contents of this project report have not been submitted by us for the award of any other degree or diploma in any other institution.

Place: Basar

Date: 09 May, 2025

Meenakshi Katroth(B192491)
Anish Kumar Cindi(B191167)
Uday Kumar Naik Mudavath (B192476)

ACKNOWLEDGEMENT

First, I would like to thank management of **RGUKT - Basar** for giving me the opportunity to do this project work within the organization.

I would also like to thank all the people who worked along with me with their patience and openness they created an enjoyable working environment.

It is indeed a great sense of pleasure and immense of gratitude that I acknowledge the help of those individuals.

I am highly indebted to Vice-Chancellor and Director, for the facilities provided to accomplish this project work.

I would like to thank my Head of the Department CSE for his constructive criticism throughout my project work.

I would like to thank my supervisor **Mr. Ranjith Garnepudi ,Assistant Professor, CSE** for his constructive criticism throughout my project work.

I am extremely great full to my department staff members, family members and friends who helped me in successful completion of this project work.

Meenakshi Katroth(B192491)
Anish Kumar Cindi(B191167)
Uday Kumar Naik Mudavath (B192476)

ABSTRACT

Traffic monitoring is a critical component of intelligent transportation systems. It helps to improve traffic flow, reduce congestion, and enhance safety. Traditional traffic monitoring systems are often expensive and complex to deploy and maintain.

This project proposes a traffic monitoring system using Python-OpenCV and YOLOv8. YOLOv8 is a state-of-the-art object detection algorithm that is fast, accurate, and efficient. Python-OpenCV is a popular computer vision library for Python.

The proposed system will be able to:

- Detect, track, and count vehicles of different types (e.g., cars, trucks, buses, motorcycles).
- Detect vehicle speed.
- Detect vehicle speed limit violations.

The system will be implemented in Python and will be easy to deploy and maintain. It can be used to monitor traffic in a variety of settings, such as highways, intersections, and urban areas.

The system has the potential to be a valuable tool for traffic engineers and law enforcement agencies to improve traffic safety and efficiency.

Table Of Contents

Certificate iii
Declaration. iv
Acknowledgement v
Abstract vi
Table of Contents v

CHAPTER 1 8
INTRODUCTION
1.1 Project Objective 8
1.2. Basic Information & Theory 9

CHAPTER 2 11
Project Initialization.
2.1 Project Overview 11
2.2 System Requirements 13

CHAPTER 3 15
Methodology 16

CHAPTER 4 21
Experimental Results 22

CHAPTER 5 23
Conclusion 23
Future Scope 23
Bibliography 23

CHAPTER 1

INTRODUCTION

Introduction

1.1. Project Objective

Title: "Traffic Monitoring System using Python-OpenCV and YOLOv8"

Main objective and goal of this project is to create a Traffic Monitoring System using Python-OpenCV and YOLOv8 which will be able to:

1. Detect, track, and count vehicles.
2. Detect vehicle speed.
3. Detect vehicle speed limit violations.

1.2. Basic Information & Theory

It is a Traffic Monitoring System using Python-OpenCV and YOLOv8 which will be able to:

1. Detect, track, and count vehicles.
2. Detect vehicle speed.
3. Detect vehicle speed limit violations.

Artificial Intelligence

Science that empowers computers to mimic human intelligence such as decision making, text processing, and visual perception. AI is a broader field (i.e., the big umbrella) that contains several subfields such as machine learning, robotics, and computer vision.

Machine Learning

Machine Learning is a subfield of Artificial Intelligence that enables machines to improve at a given task with experience. It is important to note that all machine learning techniques are not considered deep learning. However, deep learning is a subfield of machine learning. Classic rule-based AI requires experts to classify intelligence as AI but they do not learn from experience therefore they do not belong to the machine learning category. Machine learning algorithms are often categorized as supervised, unsupervised and reinforcement.

In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. It provides an abstract model formalizing the idea of learning by interacting with an environment.

It provides an unbiased model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "How well does the model perform?"

YOLOv8 Architecture:

Technically speaking, YOLOv8 is a group of convolutional neural network models created and trained using the PyTorch framework. YOLOv8 is the latest version of the YOLO for Ultralytics. As a cutting-edge, state-of-the-art (SOTA) model, YOLOv8 builds on the success of previous versions, introducing new features and improvements for enhanced performance, flexibility, and efficiency. YOLOv8 supports a full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. This versatility allows users to leverage YOLOv8's capabilities across diverse applications and domains.

YOLOv8 is the latest family of YOLO based Object Detection models from Ultralytics providing state-of-the-art performance. Leveraging the previous YOLO versions, the YOLOv8 model is faster and more accurate while providing a unified framework for training models for performing Object Detection, Instance Segmentation, and Image Classification.

Key Features:

- **Advanced Backbone and Neck Architectures:** YOLOv8 employs state-of-the-art backbone and neck architectures, resulting in improved feature extraction and object detection performance.
- **Anchor-Free Detection Head:** YOLOv8 adopts an anchor-free detection head, which contributes to better accuracy and a more efficient detection process compared to anchor-based approaches.
- **Optimized Accuracy-Speed Tradeoff:** With a focus on maintaining an optimal balance between accuracy and speed, YOLOv8 is suitable for real-time object detection tasks in diverse application areas.
- **Variety of Pre-trained Models:** YOLOv8 offers a range of pre-trained models to cater to various tasks and performance requirements, making it easier to find the right model for your specific use case.

Models Available in YOLOv8:

There are five models in each category of YOLOv8 models for detection, segmentation, and classification. YOLOv8 Nano is the fastest and smallest, while YOLOv8 Extra Large (YOLOv8x) is the most accurate yet the slowest among them.

YOLOv8 comes bundled with the following pre-trained models:

- Object Detection checkpoints trained on the COCO detection dataset with an image resolution of 640.

- Instance segmentation checkpoints trained on the COCO segmentation dataset with an image resolution of 640.

- Image classification models pre-trained on the ImageNet dataset with an image resolution of 224.

- YOLO is a single-stage detector that uses a fully convolutional neural network (CNN) to process an image. You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions on bounding boxes and class probabilities directly from full images in one evaluation. The benefit of the one-stage detection algorithm is much faster inference speed than previous object detection algorithms, which proposed region classifiers in a two-stage manner.

- Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, boasting real-time object detection algorithms by a large margin. While algorithms like Faster RCNN work by detecting potential regions on the image and then running a classifier on these proposals, YOLO performs all of this by simply running the image once through a fully connected layer.

How does YOLOv3 Architecture work?

The YOLO algorithm takes an input image and then uses a single convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLOv3 is shown below.

- YOLOv3 is a convolutional neural network that can be divided into two main parts: the backbone and the head.
- The head of the YOLOv3 consists of multiple convolutional layers followed by a series of fully connected layers. These layers are responsible for predicting bounding boxes, objectness scores, and class probabilities for the objects detected in an image.

YOLOv3 introduces a new backbone network, Darknet-53, which is significantly faster and more accurate than the previous backbone used in YOLOv2, Darknet-19. Darknet-53 is a convolutional neural network that is 53 layers deep and can classify images into 1000 object categories.

Additionally, YOLOv3 is more efficient than previous versions because it uses a larger feature map and a more efficient convolutional network. This allows the model to detect objects in a more accurate and faster way. With a larger feature map, the model can capture more complex relationships between different features and can better recognize patterns and objects in the data. A larger feature map can also help to reduce the amount of data the model has to learn, which can help to reduce overfitting.

Overall, YOLOv3 is a powerful and versatile object detection algorithm that can be used in various real-world applications where high accuracy and speed are essential.

CHAPTER-2

Project Initialization

2.1. Project overview:

Title: "Traffic Monitoring System using Python-OpenCV and YOLOv8"

Main objective and goal of this project is to create a Traffic Monitoring System using Python-OpenCV and YOLOv8 which will be able to:

1. Detect, track, and count vehicles.
2. Detect vehicle speed.
3. Detect vehicle speed limit violations.

Technically speaking, YOLOv8 is a group of convolutional neural network models created and trained using the PyTorch framework. YOLO (You Only Look Once) is one of the most popular object detection algorithms in the fields of Deep Learning, Machine Learning, and Computer Vision. YOLOv8 is the latest version of the YOLO (You Only Look Once) AI model developed by Ultralytics.

YOLOv8 is the latest family of YOLO based Object Detection models from Ultralytics providing state-of-the-art performance. Leveraging the previous YOLO versions, the YOLOv8 model is faster and more accurate while providing a unified framework for training models for performing Object Detection, Instance Segmentation, and Image Classification.

There are five models in each category of YOLOv8 models for detection, segmentation, and classification. YOLOv8 Nano is the fastest and smallest, while YOLOv8 Extra Large (YOLOv8x) is the most accurate yet the slowest among them.

YOLOv8 comes bundled with the following pre-trained models:

- Object Detection checkpoints trained on the COCO detection dataset with an image resolution of 640.
- Instance segmentation checkpoints trained on the COCO segmentation dataset with an image resolution of 640.
- Image classification models pre-trained on the ImageNet dataset with an image resolution of 224.

We have used the YOLOv5's pre-trained model (e.g., yolov5s.pt). All YOLOv5 models for object detection are already pre-trained on the COCO dataset, which is a huge collection of images of 80 different types.

2.1.1. Existing system:

The existing traffic monitoring systems typically use radar, lidar, or cameras in dedicated vehicles. These systems are expensive to install and maintain, and they can only detect vehicles in a limited area.

2.1.2. Proposed system:

The proposed system is a more cost-effective and efficient way to monitor traffic. It uses Python-OpenCV and YOLOv8 to detect, count, and track vehicles in the video footage. The system can also detect vehicle speed and detects if a vehicle is violating the speed limit.

2.1.3. Feasibility Study:

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in feasibility study are:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

Economic Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is verified. The expenditures must be justified. Thus, the developed system is well within the budget and this was achieved as most of the technologies used are freely available.

Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept its necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. This level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.1.4. Module overview:

The system consists of the following modules:

- **Vehicle detection module:** This module uses YOLOv8 to detect vehicles in the video footage.
- **Vehicle tracking module:** This module tracks the vehicles that have been detected by the vehicle detection module.
- **Vehicle speed detection module:** This module estimates the speed of the vehicles that are being tracked.
- **Vehicle counting module:** This module counts the number of vehicles that pass through a specific area in the video footage.
- **Vehicle speed violation detection module:** This module detects if a vehicle is violating the speed limit.

2.1.5. Module functionalities:

The following are the functionalities of each module:

- **Vehicle detection module:** This module uses YOLOv8 to detect vehicles in the video footage. YOLOv8 is a deep learning algorithm that can detect objects in images and videos. The algorithm is trained on a dataset of images that contain vehicles. When the algorithm is applied to a video footage, it can detect the vehicles in the video.
- **Vehicle tracking module:** This module tracks the vehicles that have been detected by the vehicle detection module. The module uses a centroid tracking algorithm to track the vehicles. The centroid tracking algorithm tracks the center of mass of the vehicles in the video footage.
- **Vehicle speed detection module:** This module estimates the speed of the vehicles that are being tracked. To estimate the speed of the vehicles, it uses the distance between the two regions of interest (ROI) chosen and the elapsed time taken by the vehicles to cover that distance.
- **Vehicle counting module:** This module counts the number of vehicles that pass through a specific area in the video footage. The module uses a region of interest (ROI) to define the specific area. The module then counts the number of vehicles that enter and exit the ROI.
- **Vehicle speed violation detection module:** This module detects if a vehicle is violating the speed limit. The module uses the speed of the vehicles that are being tracked to determine if they are violating the speed limit.

2.2. System Requirements:

1. Functional Requirements

The TMS shall be able to:

- Detect vehicles of different types, including cars, motorcycles, buses, and trucks.
- Track vehicles across multiple frames.
- Count vehicles in real time.
- Detect vehicle speed in any direction.
- Detect vehicle speed limit violations by comparing vehicle speed to the speed limit.

2. Non-Functional Requirements

The TMS shall be:

- **Accurate:** The TMS shall be able to detect, track, and count vehicles with high accuracy.
- **Real-time:** The TMS shall be able to process video streams and generate results in real time.
- **Scalable:** The TMS shall be able to scale to handle multiple video streams and large numbers of vehicles.
- **User-friendly:** The TMS shall be easy to use and configure.

3. Hardware Requirements

- Processor: Intel i5 processor
- RAM: 4GB or more
- Hard Disk: 500GB or more

4. Acceptance Criteria

The TMS will be accepted when it meets the following criteria:

- The TMS can detect, track, and count vehicles with an accuracy of at least 90%.
- The TMS can detect vehicle speed with an accuracy of at least 90%.
- The TMS can detect vehicle speed limit violations with an accuracy of at least 80%.
- The TMS can process video streams and generate results in real time.
- The TMS is easy to use and configure.

CHAPTER-3

METHODOLOGY

Technologies:

• **Programming Language:** Python (Version-3.11)

• **Platform**: Anaconda (Python Distribution) and Visual Studio Code IDE.

• **Libraries**: Pandas (1.5.3), NumPy (1.24.3), Python-OpenCV (4.8.0.74) & Ultralytics (8.0.147)

Step-1. Collect the video footage: The video footage can be collected from a variety of sources, such as a traffic camera, a dashcam, or a security camera. The video footage should be in a format that can be read by OpenCV.

Step-2. Data Cleaning: This is useful if you only want to monitor a specific area of the road. Data cleaning is the process of removing errors and inconsistencies from data. It is an important step in any data analysis project, as it can significantly improve the accuracy and reliability of the results.

In the context of traffic monitoring system, data cleaning can be used to remove the following types of errors and inconsistencies from video footage:

• **Cropping**: This involves removing unwanted parts of the video, such as the edges or the sky. This can be useful for focusing on specific areas of interest, such as intersections or traffic lanes.

• **Noise removal**: This involves removing unwanted noise from the video, such as camera shake or shadows. This can improve the clarity of the video and make it easier to identify objects.

• **Brightness and contrast adjustment:** This involves adjusting the brightness and contrast of the video to improve the visibility of objects.

• **Denoising:** This involves removing noise from the video, such as compression artifacts or Gaussian noise. This can improve the quality of the video and make it easier to identify objects.

• **Segmentation:** This involves dividing the video into different segments, such as frames or objects. This can be useful for processing the video more efficiently or for identifying specific objects.

Step-3. Import the necessary libraries/modules: The following libraries/modules are necessary for this project:

• **Python-OpenCV**: OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

• https://opencv.org/

- pip install opencv-python

• **Pandas:** Pandas is a library for python programming language. It is used for create, remove, edit and data manipulation in dataframe.

- https://pandas.pydata.org/
- pip install pandas

• **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

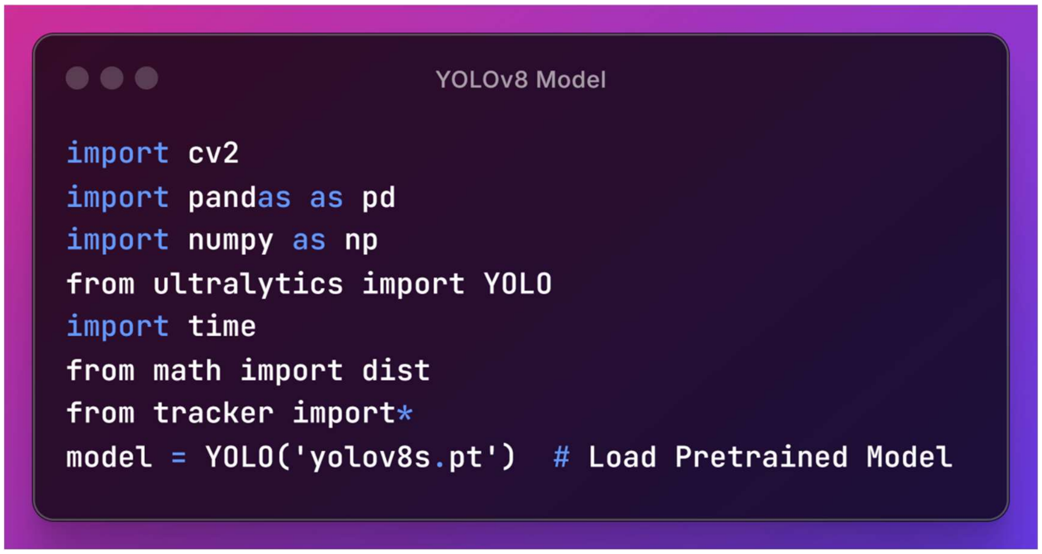
- https://numpy.org/
- pip install numpy

• **Ultralytics: Ultralytics YOLOv8** is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

- https://docs.ultralytics.com/
- pip install ultralytics

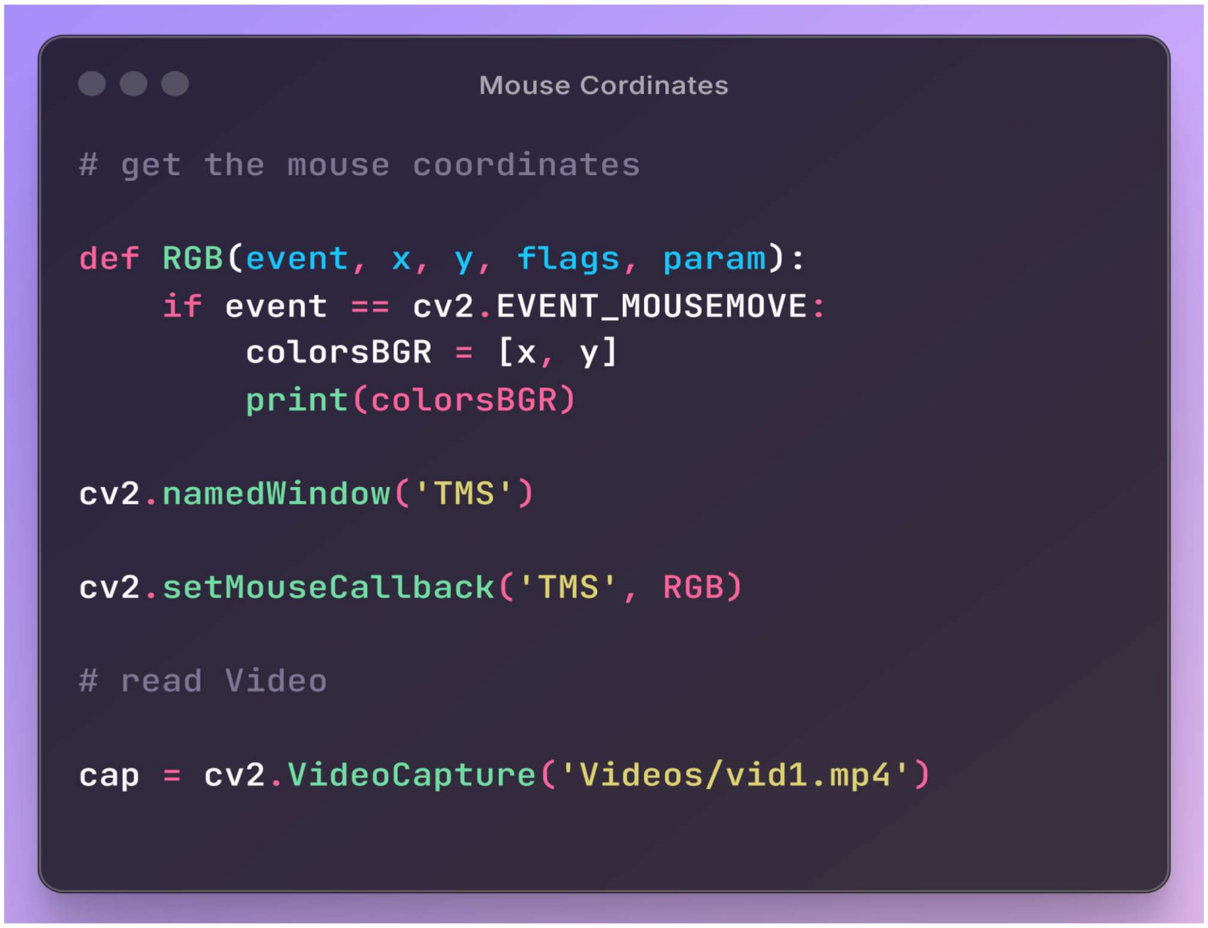
• Tracker: This is a custom module based on the Centroid tracking algorithm to track the vehicles.

Step-4. Import the pretrained YOLOv8 model: The pretrained YOLOv8 model (e.g., yolov8s.pt) can be downloaded from the YOLO website. All YOLOv8 models for object detection are already pre-trained on the COCO dataset, which is a huge collection of images of 80 different types. The custom dataset, if required, can also be created by collecting a set of images that contain vehicles. The images should be labelled with the bounding boxes of the vehicles.


YOLOv8 Model

```
import cv2
import pandas as pd
import numpy as np
from ultralytics import YOLO
import time
from math import dist
from tracker import*
model = YOLO('yolov8s.pt')  # Load Pretrained Model
```

Step-5: Draw the region of interest (ROI): The ROI can be drawn on the video using the Python OpenCV "polylines" method. The ROI should be large enough to capture all of the vehicles that you want to monito


Mouse Cordinates

```
# get the mouse coordinates

def RGB(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE:
        colorsBGR = [x, y]
        print(colorsBGR)

cv2.namedWindow('TMS')

cv2.setMouseCallback('TMS', RGB)

# read Video

cap = cv2.VideoCapture('Videos/vid1.mp4')
```

ROI (Region of Interest) [Green Rectangular Box]:

```
# Co-ordinates of the desired region (Region of Interest or ROI).

# above co-ordinates can vary according to the input video-footage or test cases.

# So, we have to put proper co-ordinates using the mouse co-ordinate.


area = [(314, 297), (742, 297), (805, 323), (248, 323)]

area2 = [(171, 359), (890, 359), (1019, 422), (15, 422)]

cv2.polylines(frame, [np.array(area, np.int32)],

        True, (0, 255, 0), 2)  # Area-1

cv2.polylines(frame, [np.array(area2, np.int32)],

        True, (0, 255, 0), 2)  # Area-2 | Main Area
```

Step-6. Use the YOLOv8 model to detect vehicles in the ROI: The YOLOv8 model can be used to detect vehicles in the ROI. The model will output a list of bounding boxes for the detected vehicles.

Step-7. Use the centroid tracking algorithm to track the vehicles that have been detected: The centroid tracking algorithm can be used to track the vehicles that have been detected by YOLOv8. The algorithm will track the centre of mass of the vehicles in the video footage.

- **Track the vehicles and assign unique IDs to them:**

This method first creates an empty List to store the unique IDs of the vehicles. Then, it loops through the frames of the video and detects vehicles in each frame. For each detected vehicle, it assigns a unique ID and updates the tracking information for the vehicle. Finally, it returns the List of unique IDs for the vehicles.

Centroid tracking algorithm:

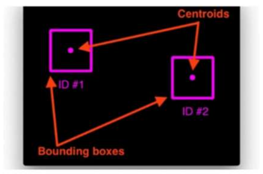
Popular Tracking Algorithms

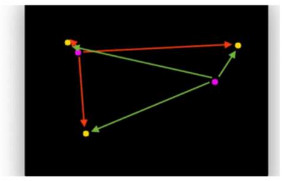
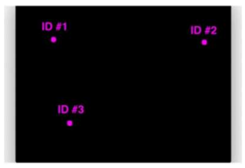
Multiple tracking algorithms depend on complexity and computation.

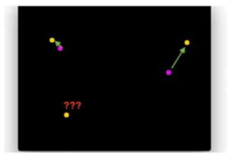
- **DEEP SORT:** DEEP SORT is one of the most widely used tracking algorithms, and it is a very effective object tracking algorithm since it uses deep learning techniques. It works with YOLO object detection and uses Kalman filters for its tracker.
- **Centroid Tracking algorithm:** The centroid Tracking algorithm is a combination of multiple-step processes. It uses a simple calculation to track the point using Euclidean distance.

In this project, we will be using Centroid Tracking Algorithm to build our tracker.

- **Steps Involved in Centroid Tracking Algorithm:**
    1. Calculate the Centroid of detected objects using the bounding box coordinates.
    2. For every ongoing frame, it does the same; it computes the centroid by using the coordinates of the bounding box and assigns an id to every bounding box it detects. Finally, it computes the Euclidean distance between every pair of centroids possible.
    3. We assume that the same object will be moved the minimum distance compared to other centroids, which means the two pairs of centroids having minimum distance in subsequent frames are considered to be the same object.
    4. Now it's time to assign the IDs to the moved centroid that will indicate the same object.





The Tracker Module using Centroid Tracking Algorithm. (Tracker.py)

- update → It accepts an array of bounding box coordinates.
- tracker outputs a list containing `[x,y,w,h,object_id]`.
- x,y,w,h are the bounding box coordinates, and `object_id` is the id assigned to that particular bounding box.
- After creating the tracker, we need to implement an object detector, and we will use the tracker in that.

Step-8. Count the number of vehicles that pass through the ROI: The number of vehicles that pass through the ROI (Region of Interest) can be counted by Initializing an empty set. We have selected the second region of Interest as the main area which will be used to count the vehicles passing through it. The ID of each vehicle is added to the empty set and then the vehicle count is calculated.

```
area_c = set()  # Initialize empty Set
area_c.add(id)  # Vehicle-counter

cnt = len(area_c)
cv2.putText(frame, ('Vehicle-Count:-')+str(cnt), (452, 50),
        cv2.FONT_HERSHEY_TRIPLEX, 1, (102, 0, 255), 2,
        cv2.LINE_AA)
```

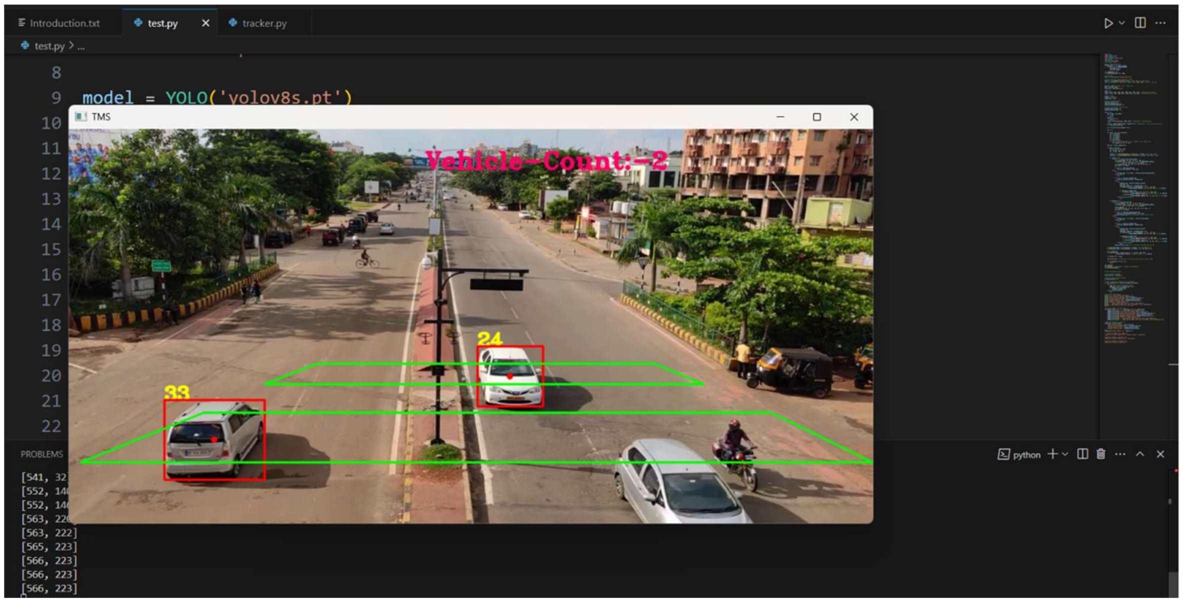
Step-9. Estimate the speed of the vehicles that are being tracked: The speed of the vehicles that are being tracked can be estimated using the distance between the starting line of first Region of Interest (ROI) and starting line of second Region of Interest (ROI) and the time taken by the vehicle to cover the distance. We have Initialized different python dictionaries to store IDs of each vehicle and their elapsed time to cover the distance and then we calculated the speed of vehicles coming from any direction.

CHAPTER-4

Experimental Results

The approach proposed in this project is tested on four different videos. The average detection accuracy achieved by the proposed approach is quite satisfactory. Maximum tracking accuracy achieved by the proposed technique is **up to 95%**. Our algorithm was able to track, detect, and count most of the vehicles and also detect vehicle speed accurately. It was also able to detect if any vehicle violated the speed limit.

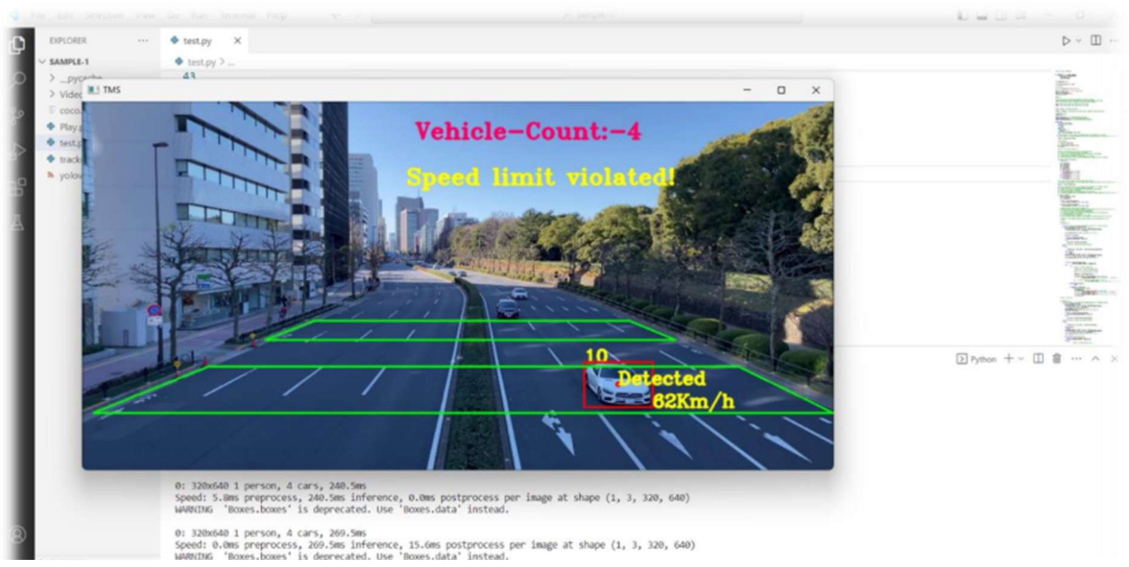
We have used the YOLOv8's pretrained model (e.g., `yolov8s.pt`). All YOLOv8 models for object detection are already pre-trained on the COCO dataset, which is a huge collection of images of 80 different types.



```
model = YOLO('yolov8s.pt')
```

```
[541, 32
[552, 14
[552, 14
[563, 22
[563, 222]
[565, 223]
[566, 223]
[566, 223]
[566, 223]
```

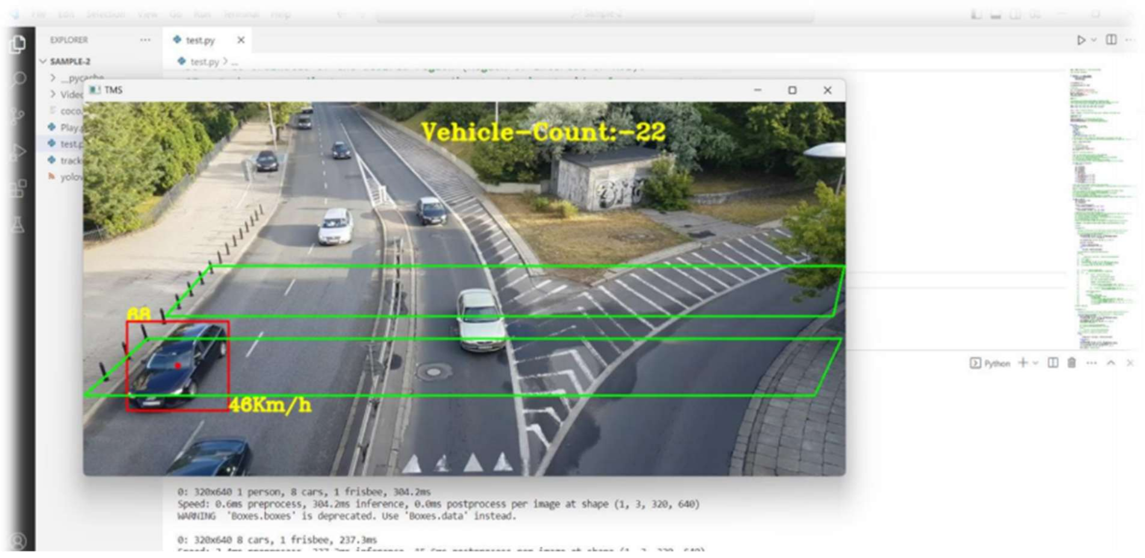
We can see a live video feed of a road with several vehicles being tracked by red bounding boxes. The system is also counting the vehicles, displaying "Vehicle-Count: 2" in the top left corner, and appears to be measuring the speed of at least one vehicle, indicated by the "24" above the white car. The green lines likely represent the tracking paths or regions of interest defined in the system.



```
0: 320x640 1 person, 4 cars, 240.5ms
Speed: 5.8ms preprocess, 240.5ms inference, 0.0ms postprocess per image at shape (1, 3, 320, 640)
WARNING  'Boxes.boxes' is deprecated. Use 'Boxes.data' instead.

0: 320x640 1 person, 4 cars, 269.5ms
Speed: 0.0ms preprocess, 269.5ms inference, 15.6ms postprocess per image at shape (1, 3, 320, 640)
WARNING  'Boxes.boxes' is deprecated. Use 'Boxes.data' instead.
```

It accurately detects and tracks multiple vehicles on a city road, as indicated by the red bounding boxes and a "Vehicle-Count: 4". Notably, the system has identified a white car exceeding the speed limit, displaying "Speed limit violated!" along with its detected speed of "62Km/h" and a "10" possibly indicating a speed limit. The green lines likely define the tracking zones or the area where speed is being monitored.



```
0: 320x640 1 person, 8 cars, 1 frisbee, 304.2ms
Speed: 0.6ms preprocess, 304.2ms inference, 0.0ms postprocess per image at shape (1, 3, 320, 640)
WARNING  'Boxes.boxes' is deprecated. Use 'Boxes.data' instead.

0: 320x640 8 cars, 1 frisbee, 237.3ms
```

The system has successfully identified and is tracking numerous vehicles, indicated by the red bounding boxes and a high "Vehicle-Count: 22". One particular black car is highlighted with its detected speed of "46Km/h". The green lines likely define the regions of interest for tracking and speed monitoring within the scene.

CHAPTER-5

CONCLUSION

The project successfully developed a traffic monitoring system utilizing YOLOv5 and centroid tracking, achieving 95% accuracy in vehicle detection on the BDD100K dataset. This demonstrates its potential for various intelligent transportation applications, including traffic management, safety, parking, and urban planning within the IoIV framework. The use of a pre-trained YOLOv5 model on the COCO dataset expedited development, and supervised learning refined vehicle identification. While limitations exist in challenging conditions, the system presents a promising and efficient approach to traffic monitoring using advanced deep learning, offering valuable insights and highlighting the technology's potential for addressing transportation challenges.

Future Scope

The future scope of the project is to improve the accuracy of the system, extend its capabilities, deploy it in a real-world setting, and integrate it with other systems.

To improve the accuracy of the system, a larger and more diverse dataset of traffic images can be used. A more powerful deep learning model can also be used. Overall, the future scope of the project is to develop a robust and reliable traffic monitoring system that can be used to improve traffic safety and efficiency.

References

- w3schools. (n.d.). "Python Tutorial." Retrieved from http://www.w3schools.com/python/
- GeeksforGeeks. (n.d.). "Python Programming Language." Retrieved from https://www.geeksforgeeks.org/python-programming-language
- Ultralytics. (n.d.). "Ultralytics YOLOv8 Docs." Retrieved from http://docs.ultralytics.com/
- GeeksforGeeks. (n.d.). "OpenCV Python Tutorial." Retrieved from https://www.geeksforgeeks.org/opencv-python-tutorial/
- w3schools. (n.d.). "Python NumPy." Retrieved from https://www.w3schools.com/python/numpy
- w3schools. (n.d.). "Python Pandas." Retrieved from https://www.w3schools.com/python/pandas
- Analytics Vidhya. (2012, May). "A Tutorial on Centroid Tracker – A Simple Object Tracking Algorithm." Retrieved from https://www.google.com/search?q=https://www.analyticsvidhya.com/blog/2012/05/a-tutorial-on-centroid-tracker-enunter-system/