UE17CS490B – Capstone Project Phase – 2

SEMESTER - VIII

END SEMESTER ASSESSMENT

Project Title : Automated Traffic Light Control And Violation Detection System
Project ID : PW21RBA03
Project Guide : Prof. Raghu BA
Project Team : PES1201700194 - Ch Abhishek
PES1201700201 - Revanth Y
PES1201700207 - Advaith K V
PES1201701667 - Mahesh H A

Outline

- Abstract
- Team Roles and Responsibilities.
- Summary of Requirements and Design (Capstone Phase - 1)
- Summary of Methodology / Approach (Capstone Phase - 1)
- Design Description
- Modules and Implementation Details
- Project Demonstration and Walkthrough
- Results and Discussion
- Lessons Learnt
- Conclusion and Future Work
- References

Abstract

- The density of vehicles on roads is rapidly increasing which has given rise to problems like accidents, traffic and bottlenecks.
- As we all know, traffic lights can only be controlled by advanced technologies as it requires a lot of surveillance and monitoring to control the traffic.
- Due to the large number of these traffic signals it is impossible to monitor all of them. Hence we need to make an automated traffic light system that enables the flow of traffic as and when required by it.
- Road safety is a major concern by traffic as many rides fail to maintain all the traffic rules and they continue to violate them.

Abstract

- To help traffic policemen solve this problem, we need an automated system which can at the very least help the policeman and the government to curb this problem.
- We propose to build a system which can monitor and report potential violations.
- The basic methodology of our project is that we use image processing techniques for all the image recognition and measurement, the density of vehicles, speed of vehicles can be computed. Capturing images of moving vehicles can also be done.

Objective

- Our proposed system aims to achieve the following features/objectives,

  To automate and control the traffic-lights system by setting a timer.

  - To allocate time for each signal based on density.
  - To follow a specific scheduling order for traffic lights.
  - Detection of helmet when violation occurs.
  - To detect the vehicles whenever they cross the line.
  - To detect the number plate of the vehicle when a potential violation is incurred.

Team Roles and Responsibilities

Ch Abhishek	Mahesh H A	Advaith K Vasisht	Revanth Y
→ Line Violation Detection → Dataset Creation → UI Design → Flask Backend	→ License Plate Recognition → Dataset Creation → UI Design → Flask Backend	→ Automation of traffic lights → Dataset Creation → UI Design → Flask Backend	→ Helmet Detection → Dataset Creation → UI Design → Flask Backend

Summary of Requirements and Design

FUNCTIONAL REQUIREMENTS:

- **Image Recognition:** Taking the camera feed as input and splitting it into frames in order to recognize objects based on a few parameters. The output is the object frame.

- **Feature Extraction:** The frames received are fed as input and matched to the respective attribute and those attributes are sent as features for further processing.

- **Density Calculation:** The features extracted are counted and density is calculated based on the count and sent to the next phase.

Summary of Requirements and Design

Controlling traffic lights based on density: The received density is compared with a forecasting model to determine the status of the light and the new status is sent to the light.

Violation detection: The features extracted are taken as the input given by the feature extraction module and checks if a violation has occurred and returns a boolean value.

Number Plate Extraction: In case of violation signal sent by the violation detection module, the image is received and ANPR is used to recognise the number plate and the image and the number is sent for further action.

User Classes and Characteristics

Policeman:

To be able to review remotely the potential violations that have occurred. To be able to get the details of the licence plate of those violating vehicles. He/She may then confirm the violation of the rules and then impose a fine or discard the report. This prevents this user class from using a constant monitoring system and to use an asynchronous monitoring system.

Public:

Reduction in traffic during peak hours and smooth flow of intersection traffic. Prevention of jams and reduction of traffic violations. This improves the throughput of individual users and may reduce pollution due to lesser usage of vehicles.

Requirements

External Interface Requirements:

User Interfaces

- ➔ TKinter with
- ➔ Weights
- ➔ Class
- ➔ Type of violation
- ➔ Photographic evidence of Violation
- ➔ Number plate Photograph.
- ➔ Violated or Not Violated Button

Requirements

Hardware Requirements

- Camera with at least 12 MP.
- Camera Feed either wired or wirelessly.

Software Requirements

- Any Linux based OS
- Python-3
- Most recent version of Anaconda and its set of python libraries which include Scikit as well as OpenCV.

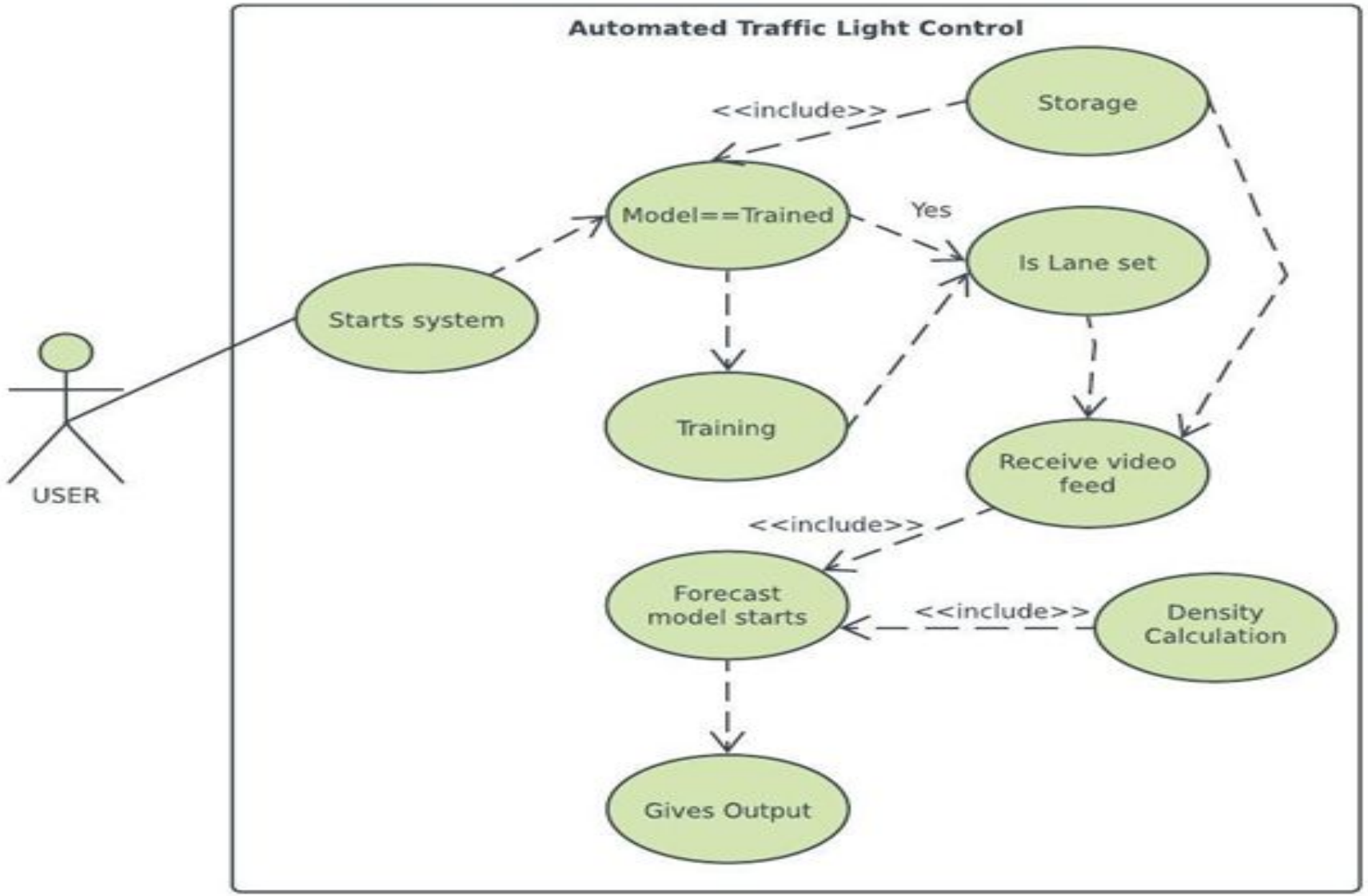
Design Approach

Benefits	Drawbacks
Image processing is low cost solution since cameras already exist at junctions	The cameras are still prone to damage and reply upon maintenance
With current ML techniques like YOLO, object and violation detection are very accurate	Even with the existing techniques, more than accuracy, the quality of image and the FOV of the camera make it harder to track violations
The difference between traditional infrared an image based recognition is that there is a lower chance of error in estimation of density	The camera based solution is heavily dependent on lighting and image enhancement techniques which may lead to slight latency

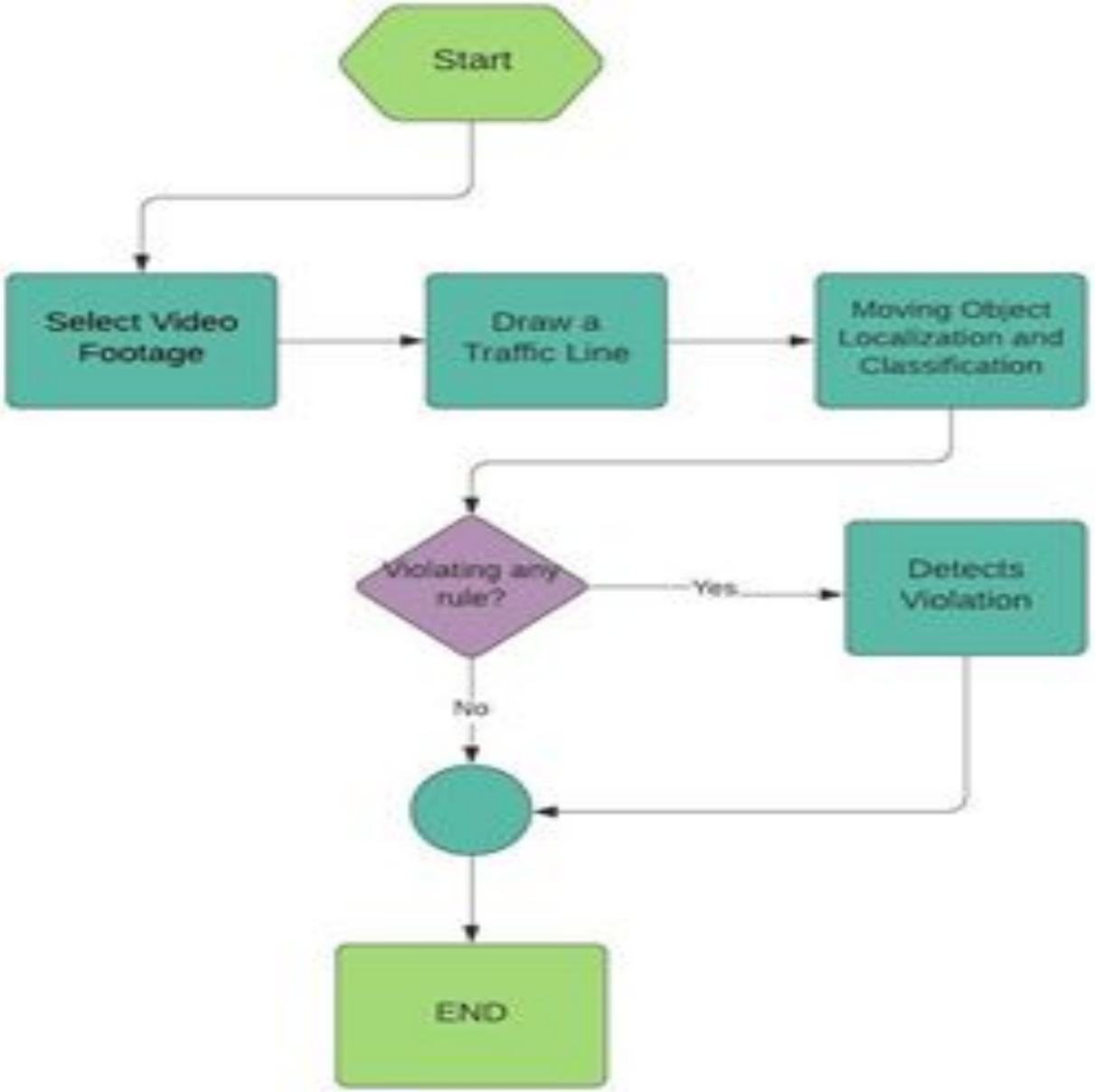
Design Constraints, Assumptions & Dependencies

- Constraints:
  - Camera Quality,
  - Availability of hardware processors
- Assumptions:
  - Cameras at every junction and with enough lighting.
  - Proper connections between the surveillance centre and the cameras.
  - Visually identifiable violations.
- Dependencies: Adequate Lighting.
  - Electricity
  - Regular Maintenance of cameras.

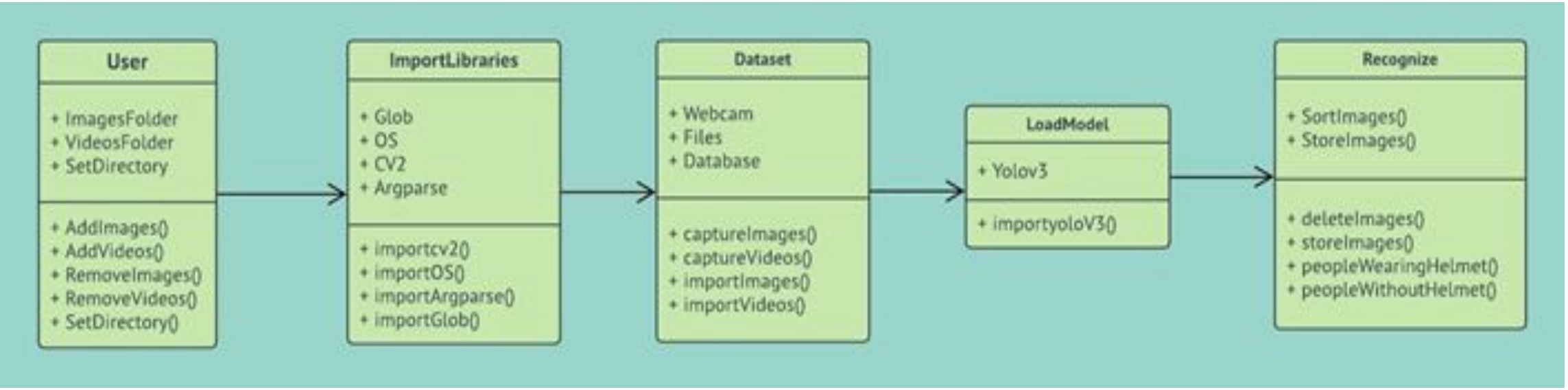
Use Case Diagram For Automated Traffic Light Control



Activity Diagram For Line Violation



Class Diagram For Helmet Detection

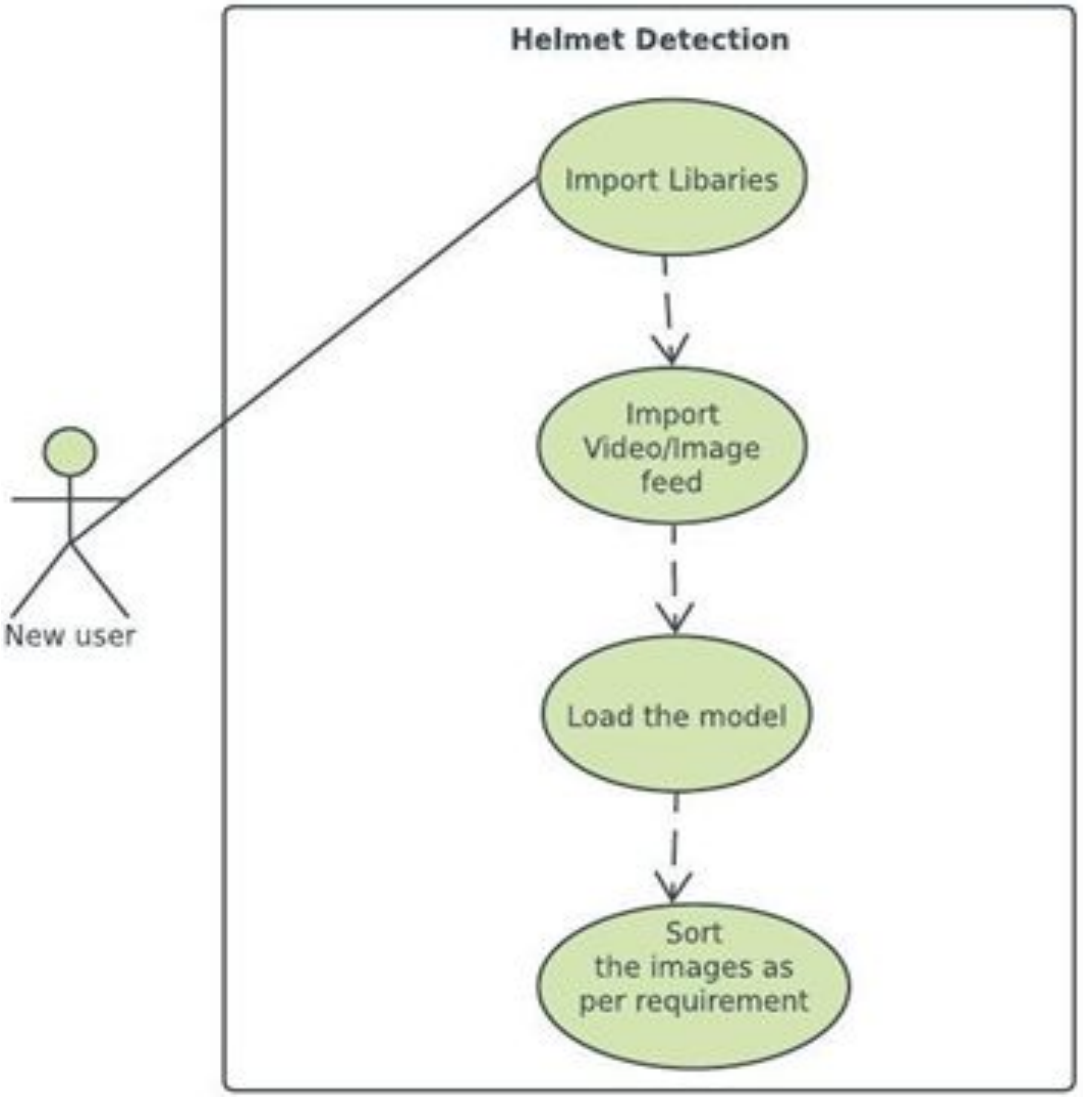


ImagesFolder/VideosFolder: They have access to the videos/images that are captured during traffic.

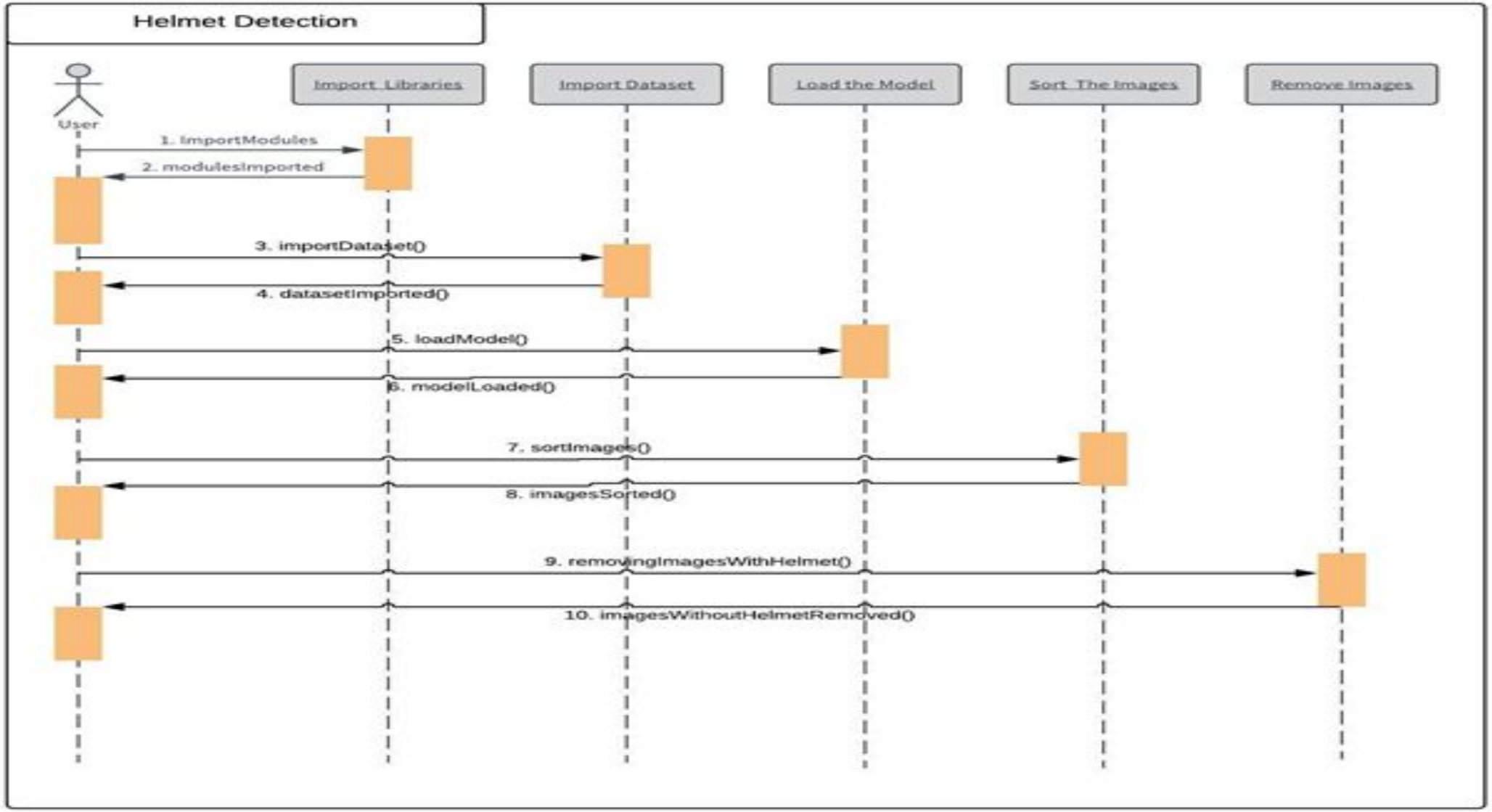
AddImages/AddVideos: The Traffic Police can feed a particular city/junction images to the helmet detection model.

RemoveImages/RemoveVideos: The Traffic Police can remove particular images/videos from the helmet detection model.

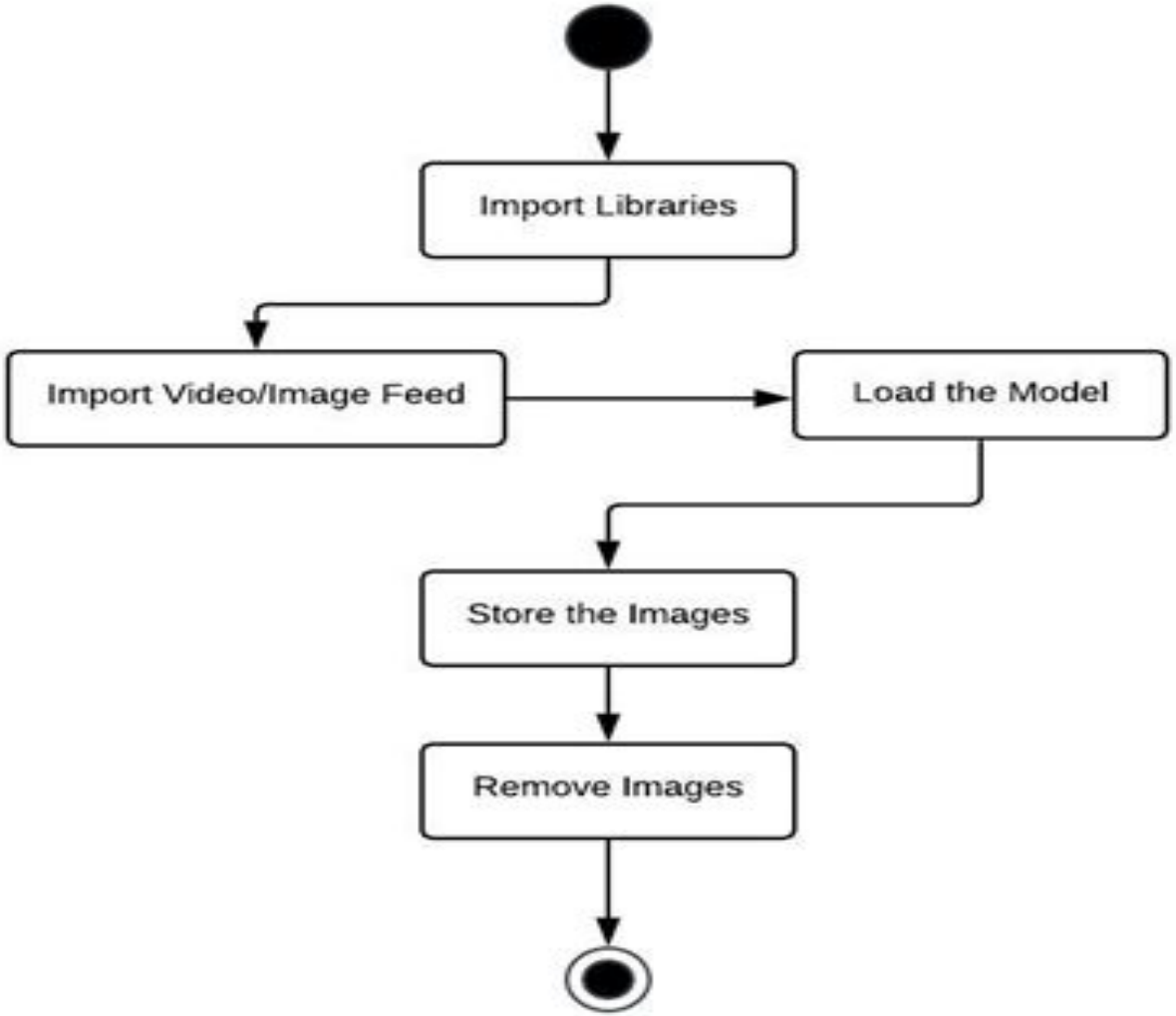
Use Case Diagram For Helmet Detection

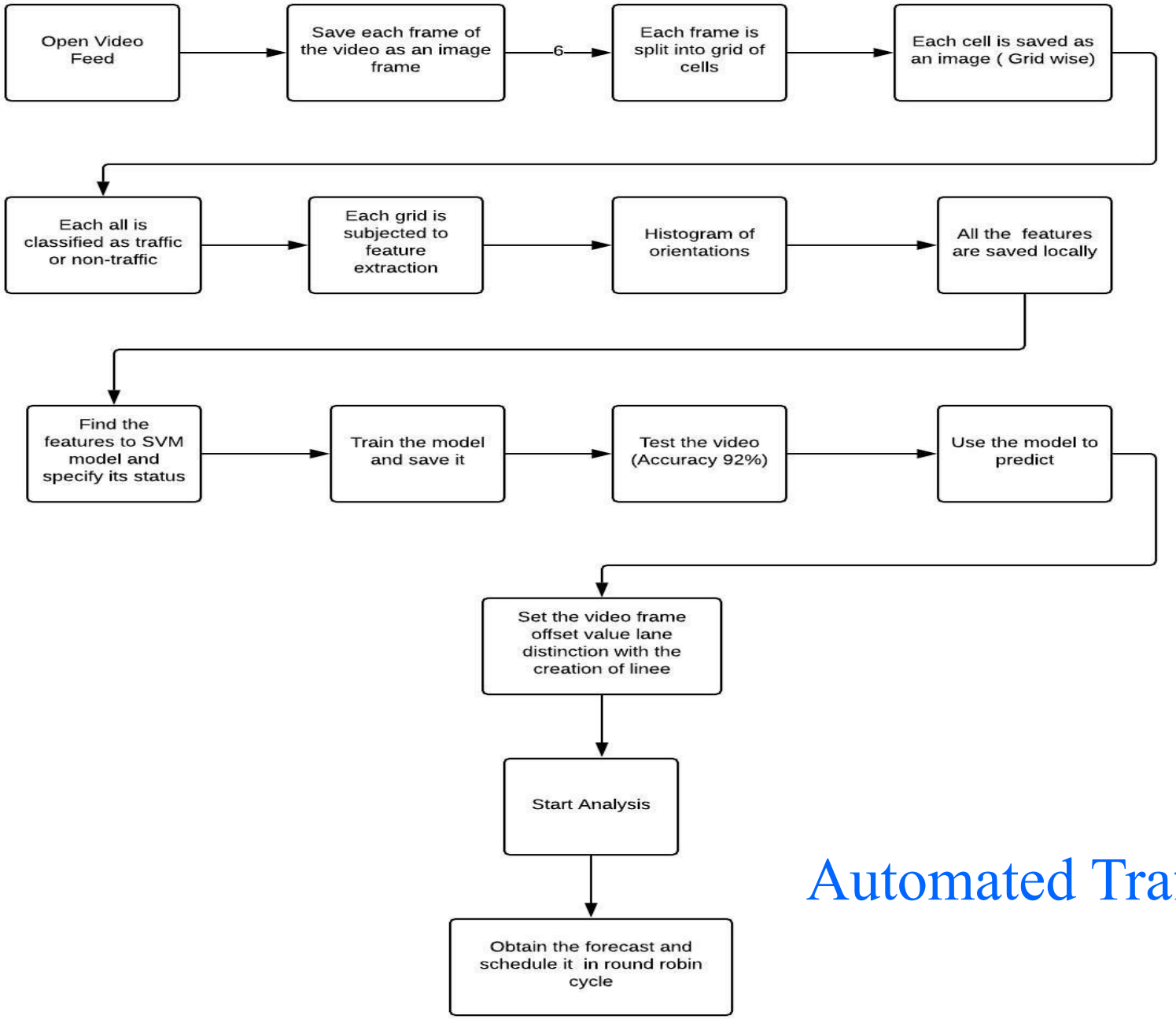


Sequence Diagram For Helmet Detection

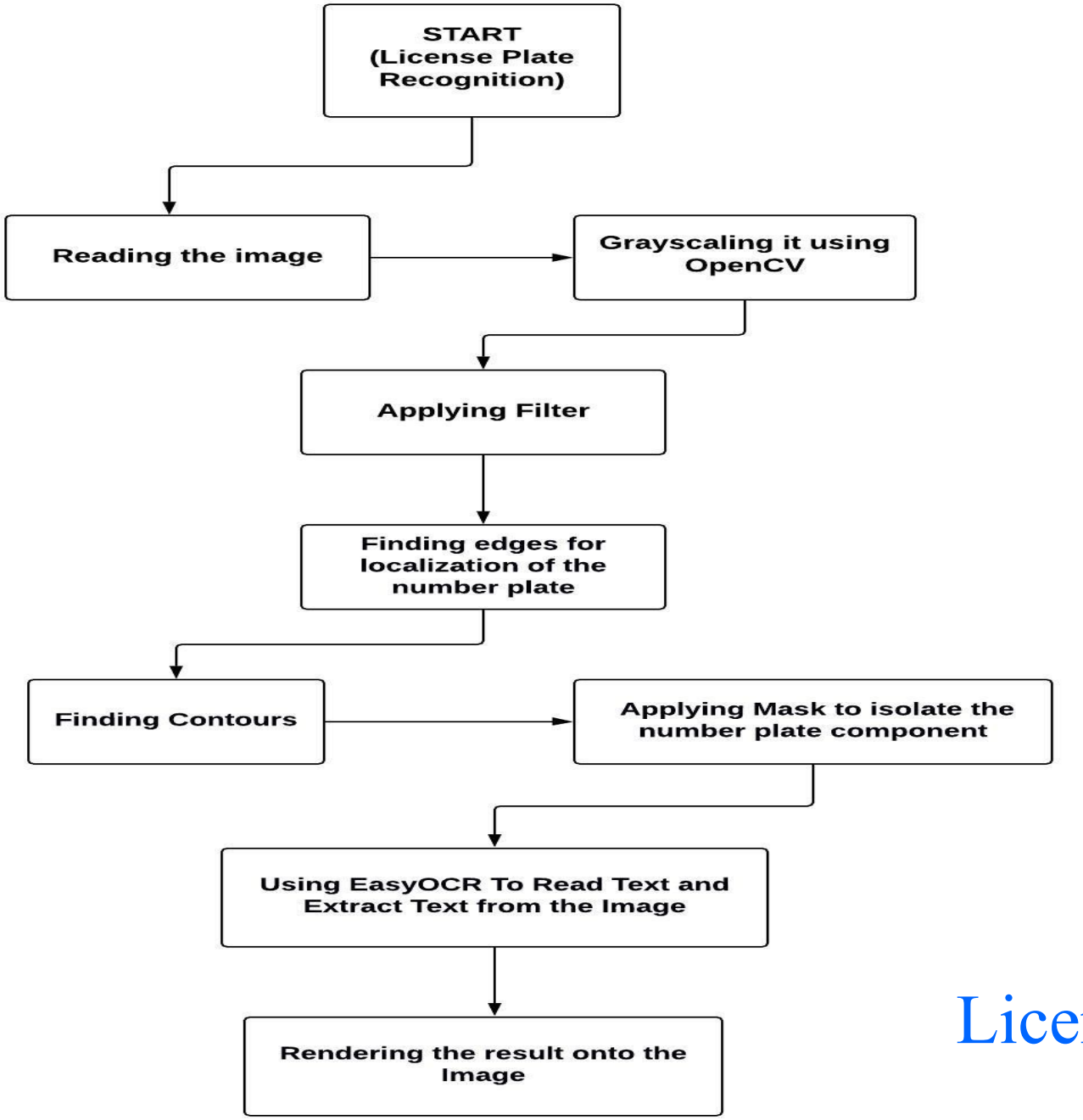


Activity Diagram For Helmet Detection





Automated Traffic Light Control WorkFlow



License Plate Recognition WorkFlow

Modules and Implementation Details

Automated Traffic Lights Control

Overview:

1) Read in a video into python using opencv
2) Split into frames and then grids and extract the ROI
3) Extract features from the ROI and load those features into a machine learning model.
4) Obtain the prediction and density and control the light based on the density.

Automated Traffic Lights Control - Implementation

- **Technology used** :- SVM, Image feature Extraction using Gradients, ARIMA
- **Libraries used** :- numpy , OpenCv , pandas, sklearn, statsmodel, pickle
- **Computer Vision** :- OpenCV is an open source computer vision and machine learning software library which is used in this project for image processing purpose.
- **Pickle:** This is the module that we use to store the python constructs and the data we are analysing.

Automated Traffic Lights Control - Implementation

Input:

- The set of training images are fed into the system by a camera located at the junction in the form of a video.
- The video is then analysed frame by frame and each frame is saved.
- The saved frame is split into grids and each grid is stored as an image locally.
- These images have to be sorted manually as we are using a training set but only once does the model have to be developed.

Automated Traffic Lights Control - Implementation

Feature Extraction:

- We the proceed to run a feature extraction algorithm on these grids individually.
- The features are then extracted and stored in a dictionary mapped to the images.
- The features extracted are then again stored locally in a serialised format to be sent to the classifier.

Automated Traffic Lights Control - Implementation

Training the model:

- We will then send the extracted features into an SVM model and train it.
- Since SVM model is based on a kernel trick, we will use a linear kernel as our data is binary data and can be separated distinctly.
- The model is now saved and the training data can be discarded.

Automated Traffic Lights Control - Implementation

Density Calculation:

- We load in our trained model and run it on the video.
- Each frame is again split into grids and the grids are fed into the model The model classifies the grid as containing vehicles or not.
- Each of the grids containing a vehicle are marked in red and the rest in green.
- The density is obtained by number of grids containing vehicle/ total number of grids.

Automated Traffic Lights Control - Implementation

Light Control:

- Now that we have the density, we will record the first 10 densities.
- The window is is 10 currently and can be expanded. After the first ten densities are calculated, we send these records to a moving average model since it is a time series. We have used ARIMA in our project.
- The ARIMA model now forecasts the next density and based on that, the green light duration is decided as follows

Automated Traffic Lights Control - Implementation

Light Control(Contd..):

Density(%)	Green Light Duration(sec)
0-30	15
30-50	20
50-70	30
70-90	45
90-100	60

Line Violation - Implementation

- **Technology used** :- YOLOv3
- **Libraries used** :- numpy , OpenCv , PIL , tkInter , imageio , Keras
- **Computer Vision** :- OpenCV is an open source computer vision and machine learning software library which is used in this project for image processing purpose.
- **PIL:** This module adds additional functionality like opening, manipulating, and saving many different image file formats.

Line Violation - Implementation

YOLO Algorithm:

Basic Idea :

- Divide the input image into grids of equal size (sxs)
- Predict a class and a bounding box of objects present in the grid for every location.

YOLO Features :

- Computationally very fast, can be used on real time environment.
- Globally processing the entire image once only with the CNN
- Maintains a high accuracy range.

Line Violation - Implementation

How it works:

1) We split the image into SxS grid.

2) Grid cell responsible for detecting an object

3) If the center/midpoint of an object falls into the grid cell, that cell is responsible for detecting the objects.

4) Each cell predicts B number of bounding boxes with a confidence score for each of the boxes.

Confidence score means how confident the model is that

- The box contains an object
- How accurately the box has predicted the object boundaries

Line Violation - Implementation

Training

1 - pedestrian
2 - car
3 - motorcycle

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

y is $3 \times 3 \times 2 \times 8$

Line Violation - Implementation

- The vehicles are detected using YOLOv3 model. After detecting the vehicles, violation cases are checked.
- A traffic line is drawn over the road in the preview of the given video footage by the user.
- The line specifies that the traffic light is red. Violation happens if any vehicle crosses the traffic line in red state.

Line Violation - Implementation

- The training images are obtained from a traffic camera located at a junction.
- The detected objects have a green bounding box. If any vehicle passes the traffic light in red state.
- If violation happens. After detecting violation, the bounding box around the vehicle becomes red and capture the vehicle which is violated.

Helmet Detection - Implementation

Detection of non-helmet riders using YOLO algorithm :

- **Technology used :-** YOLOv3
- **Libraries used :-** Numpy , Argparse , Glob , OpenCv , SYS , OS
- **drawPred** function is used to predict bounding boxes on the given frame.
- **Postprocess** function is used to remove the bounding boxes with low confidence using non-maxima suppression

Helmet Detection - Implementation

- Finally we are having the iterator through input images feed where we have to identify the persons who are not wearing helmets.
- In that we are creating blob frame
- Sending the input feed to the network
- Runs the forward pass to get output of the output layers
- By Calling Post Process function we are Removing the bounding boxes with low confidence and retaining the bounding boxes which having high confidence.

Helmet Detection - Implementation

Detection of non-helmet riders using YOLO algorithm :

- The result of calling post process function which classifies the images into two parts like the person wearing helmet and the person not wearing helmet.
- For Further reference purpose we are storing the both type of classified images.

Helmet Detection - Implementation

- The person who’s not wearing helmet will be treated as violation.

- Then we are doing Number Plate recognition on the particular image to find the license plate of the violated vehicle.

License Plate Recognition - Implementation

Overview:

1) Read in an image into python using opencv
2) Apply filtering, edge detection use contour search to find plates
3) Extracting number plate text using ocr with easy ocr

License Plate Recognition - Implementation

Steps:

1) Install and import dependencies (Modules) :

Easyocr : EasyOCR is a python package that allows the image to be converted to text.

Imutils : Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

License Plate Recognition - Implementation

2) Read in Image, Grayscale and Blur:

- Initially image is loaded using imread.
- Next we use cvtColor to recolor the image.
- Then, convert it into grey image that is done by passing the color code converter cvtColor(gray, cv2.COLOR_BGR2GRAY).
- So here we change the image color which is initially in BGR to GREY.

License Plate Recognition - Implementation

3) **Apply filter and find edges for localization**

- In here we are applying filtering to remove noise from the image and edge detection to find the edges in the image. For filtering we use bilateral filter method to that we pass the image and pass the properties which allow us to specify how intensely we want the noise reduction.

- Then from that we use canny algorithm to perform edge detection which has set of parameters which can be tuned to get the ideal edges.

License Plate Recognition - Implementation

4) Find Contours and Apply Mask:

- We do contours detection that is detecting the polygons within the lines.
- First step we do is to find contours by using the method findContours to which we pass the edge image. And here we are approximating the required contour value by using cv2.CHAIN_APPROX_SIMPLE.
- And then we store the contours by using function grab_contours and we sort the top 10 contours on basis of contour area.
- Next we loop through these contours and find whether they are rectangle or square shaped. And by doing that we get the coordinates to the number plate

License Plate Recognition - Implementation

4) Find Contours and Apply Mask (Contd..):

- Next we mask and isolate the section of number plate using the coordinates.
- Here first we masked the entire grey image. And next we highlight only number plate using drawContours function where it is given location of the number plate as parameter to it.
- And in final step we bitwise_and to put up the number plate on to the masked image.
- Next we isolate the segment which contains the number plate by cropping that part.
- So here we find the coordinates of every section where our image is in black and crop the image.

License Plate Recognition - Implementation

5. **Using Easy OCR To Read Text:**
    - In this step we use easy ocr to read the text from the image.

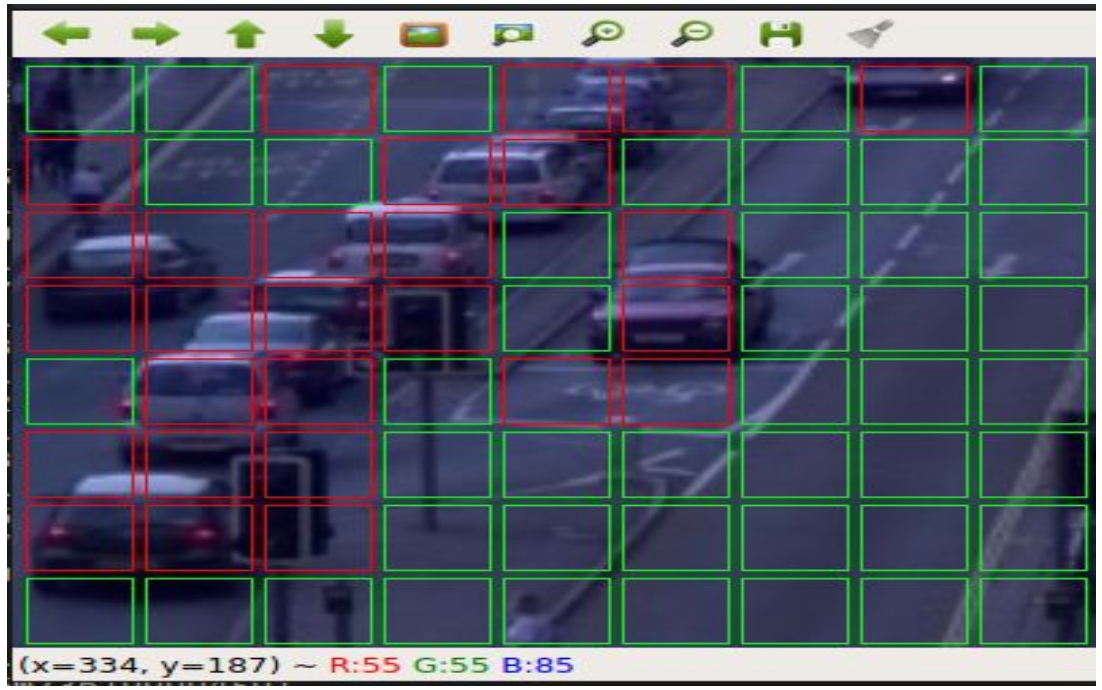
6. **Rendering The Result:**
    - In this part we take our number plate and overlay our detection on the original images.

Design Approach

Benefits	Drawbacks
Image processing is low cost solution since cameras already exist at junctions	The cameras are still prone to damage and reply upon maintenance
With current ML techniques like YOLO, object and violation detection are very accurate	Even with the existing techniques, more than accuracy, the quality of image and the FOV of the camera make it harder to track violations
The difference between traditional infrared an image based recognition is that there is a lower chance of error in estimation of density	The camera based solution is heavily dependent on lighting and image enhancement techniques which may lead to slight latency

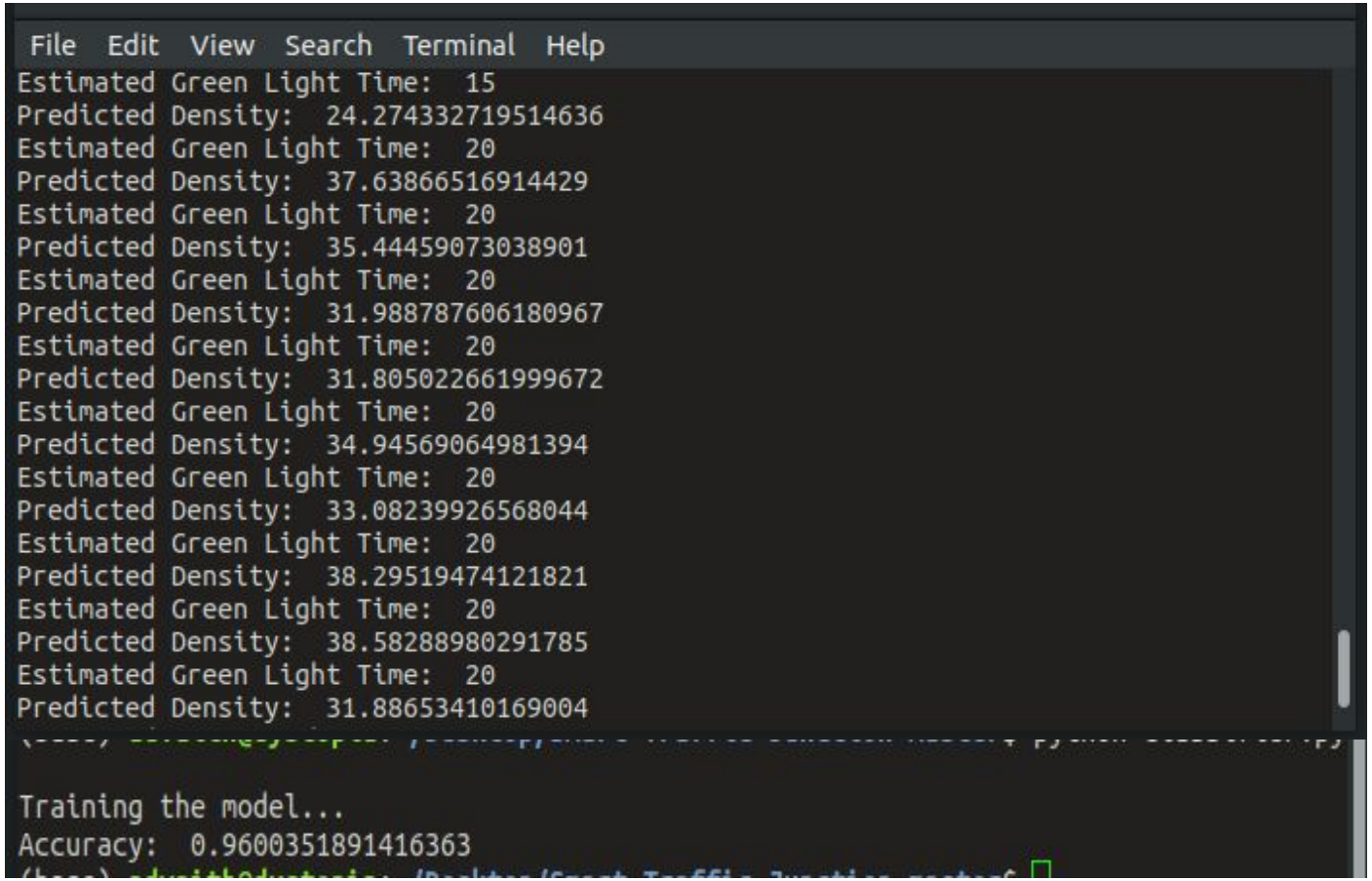
DEMO PRESENTATION

Traffic Light Automation - Results



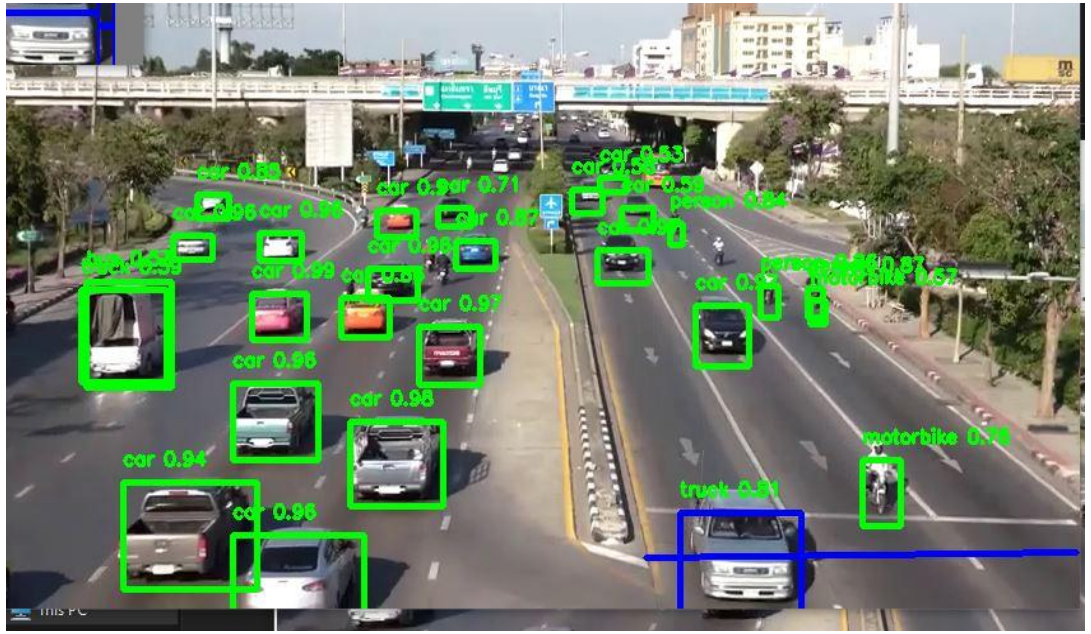
- This is how the result appears after the grid split and after the model has classified the grid.
- The red coloured grids are supposed to be grids containing traffic vehicles and the green coloured grids are ones that do not contain.

Traffic Light Automation - Results



The model gives us an accuracy of 90-100 range now. For each frame, the density is stored and then forecasted. And after each time interval, the predicted density and green light time is displayed.

Line Violation - Results

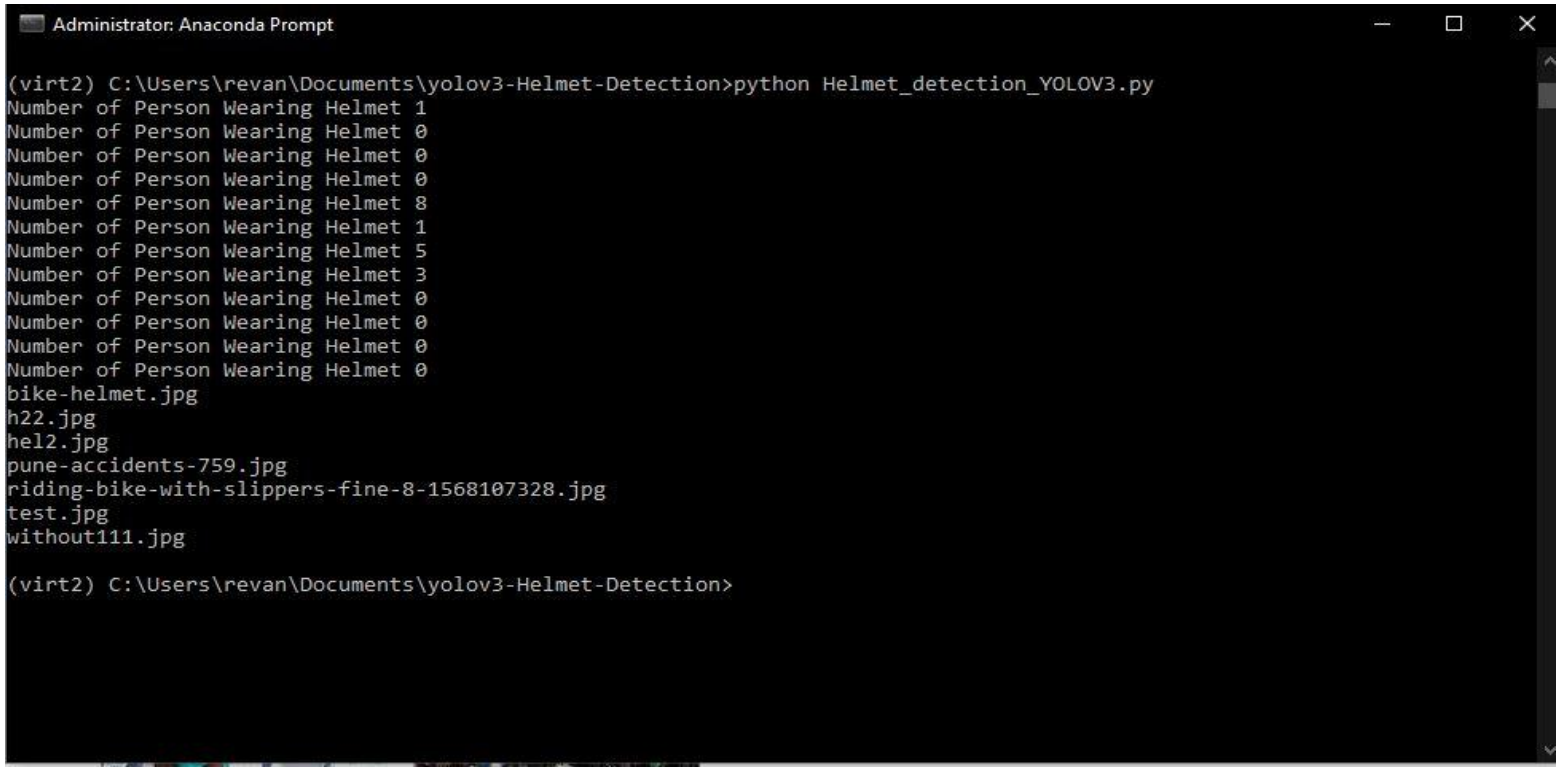


- In the below image the moving objects are detected.
- An object detection model YOLOv3 is used to classify these moving objects into respective classes.
- After detecting the vehicles, violation cases are checked.
- A traffic line is drawn over the road in the preview of the given video footage by the user. The line specifies that the traffic light is red.

Line Violation - Results

- If any vehicle passes the traffic light in red state violation happens.
- The image of the vehicle is captured and stored.
- And these images are sent as an input to License Plate Detection Model.

Helmet Detection - Results



The image below gives information about the number of persons wearing helmets and the captured images of people not wearing any helmet.

Results and Discussion

Riders who are not wearing helmets are violating the rule and these violated vehicles captured images are stored under a detected images folder. And these images are sent as an input to License Plate Detection Model.

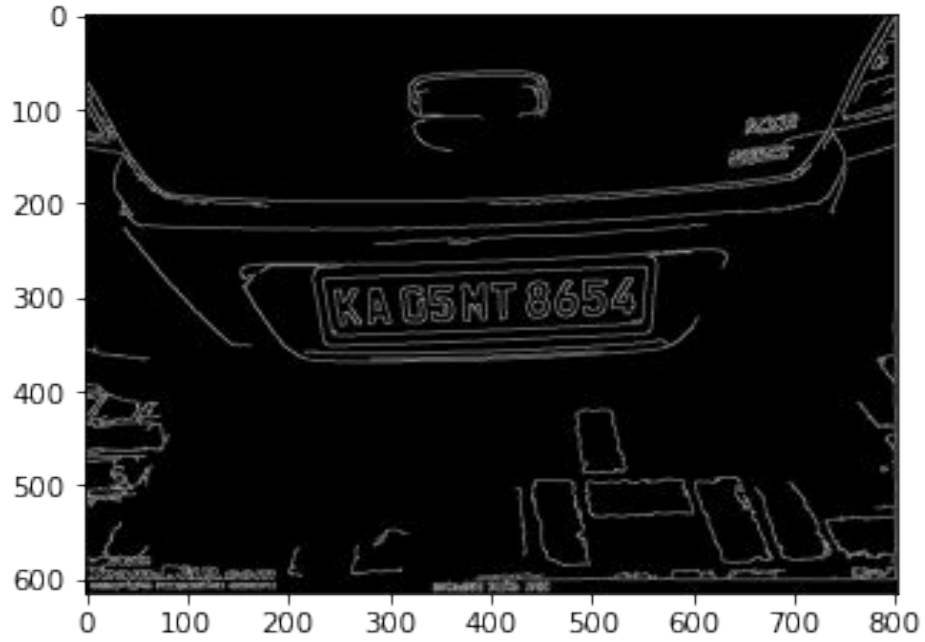
Helmet Detection - Results

In the above image, we can see people wearing helmets are detected and classified in helmet class with the confidence value shown

License Plate Recognition - Results

Reading the Image and Gray scaling it using opencv

License Plate Recognition - Results



Applying filter and finding edges for localization of the number plate

License Plate Recognition - Results

Finding Contours and Applying Mask to isolate the number plate component

License Plate Recognition - Results

Using EasyOCR To Read Text and Extract Text from the Image

License Plate Recognition - Results

Rendering the result onto the Image

Conclusion

- In order to overcome the problems faced by the traffic police and the public regarding the traffic control system, we have come up with a better solution that uses image processing techniques, scheduling algorithms which is possibly optimal and better than many other existing systems.

- We have built a system that detects the potential traffic violations and automates the traffic light signal controller. By using all the actual real time images as an input, this method is consistent in its own ways of vehicle detection and also much better than already existed systems.

Conclusion and Future work

.

The project can be improved in a lot of aspects and more work can be put into place such as a report can be generated to the vehicles which violate the traffic rules with detailed information about the rule violated, the place and the video footage for reference. And we should make sure the website should be fault tolerant.

References

[1] Pandit, V., Doshi, J., Mehta, D., Mhatre, A. and Janardhan, A., 2014. Smart traffic control system using image processing. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *3*(1), pp.2278-6856

[2] Khanke, P. and Kulkarni, P.S., 2014. A technique on road traffic analysis using image processing. *International Journal of Engineering Research and Technology*, *3*, pp.2769-2772.

[3] Soman, R. and Radhakrishnan, K., 2018. Traffic Light Control and Violation Detection Using Image Processing. *Traffic*, *8*(4).

[4] Narkhede, A., Nikam, V., Soni, A., Sathe, A. and Chiddarwar, G.G., Automatic Traffic Rule Violation Detection and Number Plate Recognition

[5] Zaatouri, K. and Ezzedine, T., 2018, December. A Self-Adaptive Traffic Light Control System Based on YOLO. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)* (pp. 16-19). IEEE.

[6] *Saha, Satadal & Basu, Subhadip & Nasipuri, Mita & Basu, Dipak. (2009). Development of an automated Red Light Violation Detection System (RLVDS) for Indian vehicles.*

Thank You