National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

ROBOTICS – I (GROUP 1)
Project Report

TRAFFIC MANAGEMENT VIA COMPUTER VISION

TEAM MEMBERS (BEE-11A)

Name	CMS ID
Fatima Iqbal	290358
Daniya Imran	321940
Haleema Ushaq	291196
Hamzah Sohail	286248
Ahmed Munir	321653
Yousaf -	321798

Introduction

Computer vision can be used to analyze video feeds from traffic cameras to detect the number of vehicles at intersections and optimize traffic signal timing accordingly. Our idea isto design a 4-way traffic control system that uses Computer Vision in its decision-making process and can detect any anomalies such as traffic violation and speed detection.

Modules

Our project has 3 modules:

Module 1: Traffic Light Control

Computer vision can be used for traffic violation control by analyzing video footage of vehicles to detect when they are breaking traffic laws. For example, computer vision algorithms can be used to detect when a vehicle is running a red light, speeding, or making anillegal turn. This can be done by training a machine learning model to recognize these typesof violations in video footage and then using that model to analyze video feeds from traffic cameras or other sources.

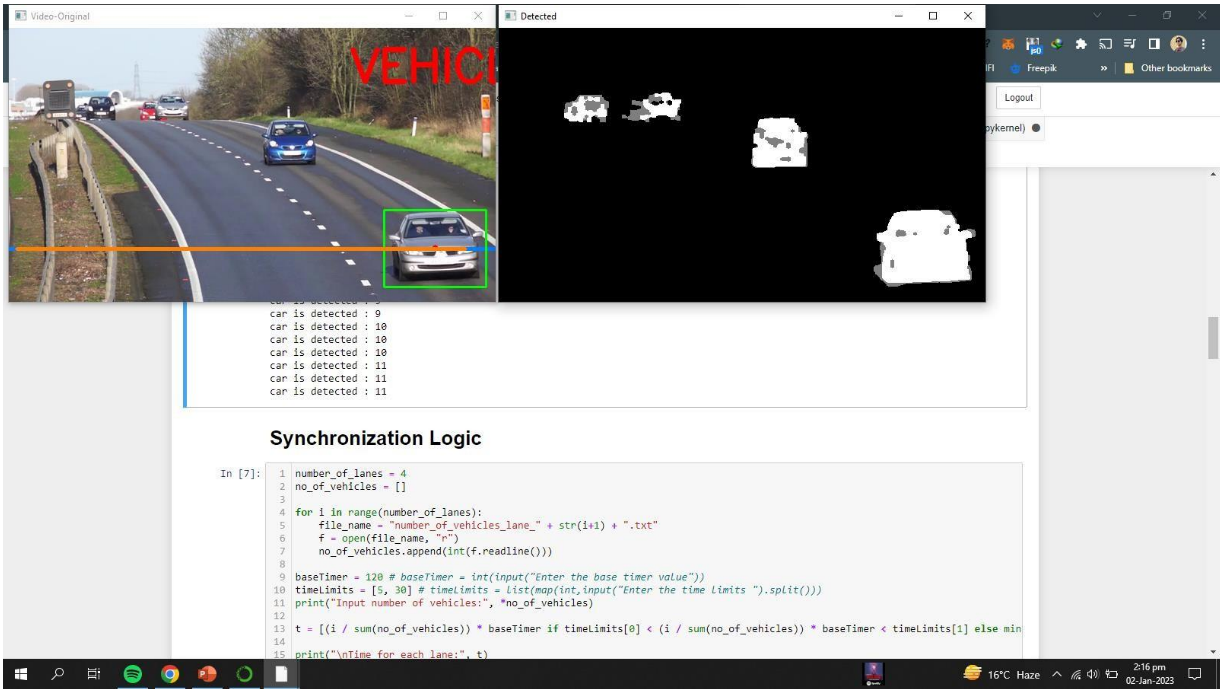
Design details:

A camera will be used detect the number the cars in each lane on a road going towards the intersection. After receiving image form camera (as shown above), we will go into the processing stage where we will use OpenCV to calculate the number of cars in each lane at the intersection in real time.

This data will then be fed through a Probabilistic Algorithm which will then decide which signal to give. Our module was able to count the number vehicles passing through each lane and change the traffic lights changing time accordingly and detect any traffic violations.

Results:

Vehicle Counter

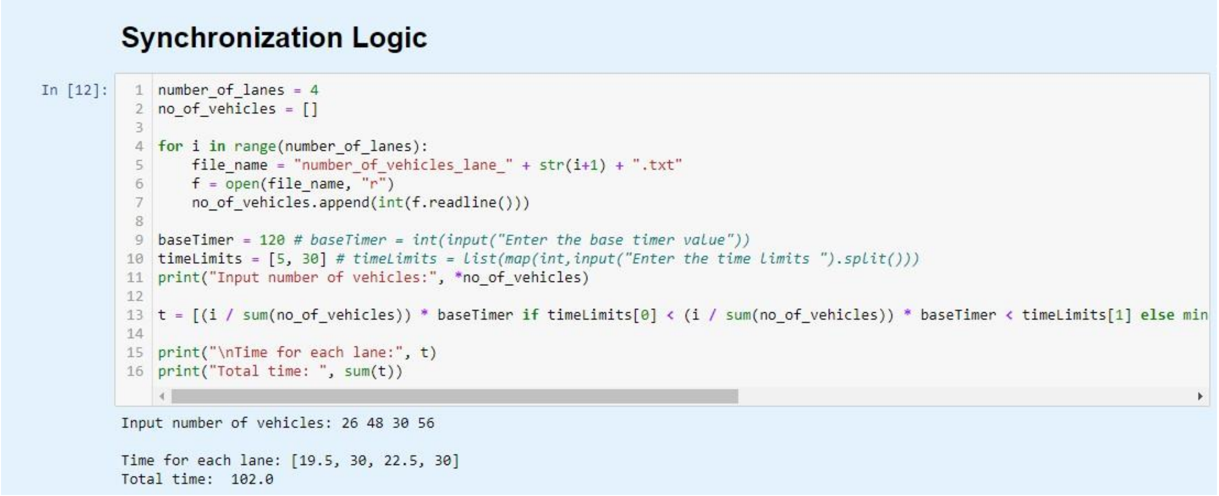


```
car is detected : 9
car is detected : 10
car is detected : 10
car is detected : 10
car is detected : 11
car is detected : 11
car is detected : 11
```

Synchronization Logic

```
In [7]:
number_of_lanes = 4
no_of_vehicles = []

for i in range(number_of_lanes):
    file_name = "number_of_vehicles_lane_" + str(i+1) + ".txt"
    f = open(file_name, "r")
    no_of_vehicles.append(int(f.readline()))

baseTimer = 120 # baseTimer = int(input("Enter the base timer value"))
timeLimits = [5, 30] # timeLimits = list(map(int,input("Enter the time limits ").split()))
print("Input number of vehicles:", *no_of_vehicles)

t = [(i / sum(no_of_vehicles)) * baseTimer if timeLimits[0] < (i / sum(no_of_vehicles)) * baseTimer < timeLimits[1] else min

print("\nTime for each lane:", t)
```

Calculating the Time for Traffic Signal (Synchronization Logic):



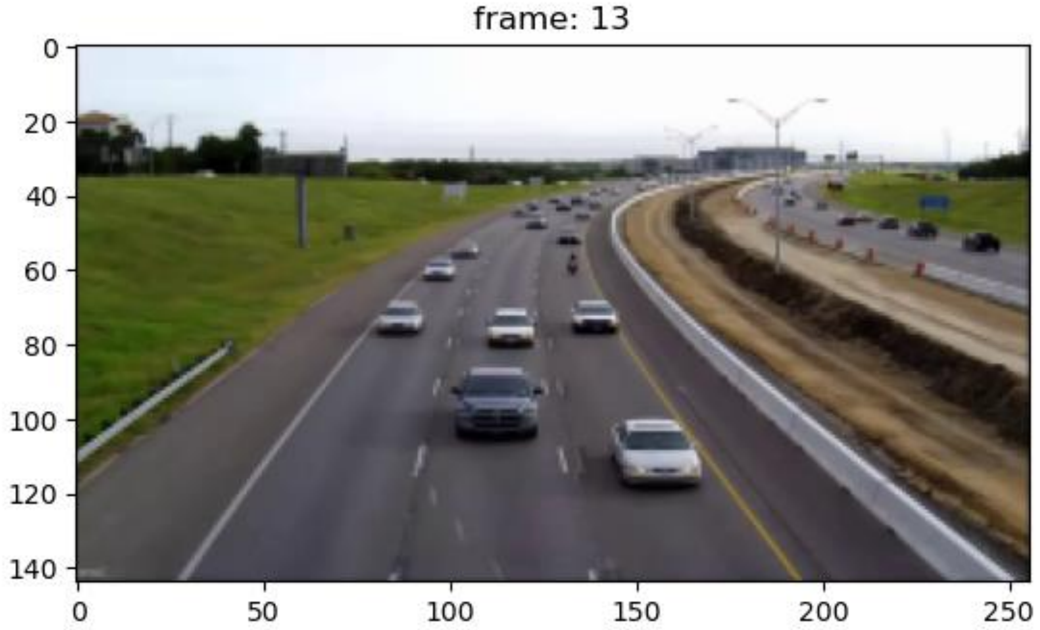
Synchronization Logic

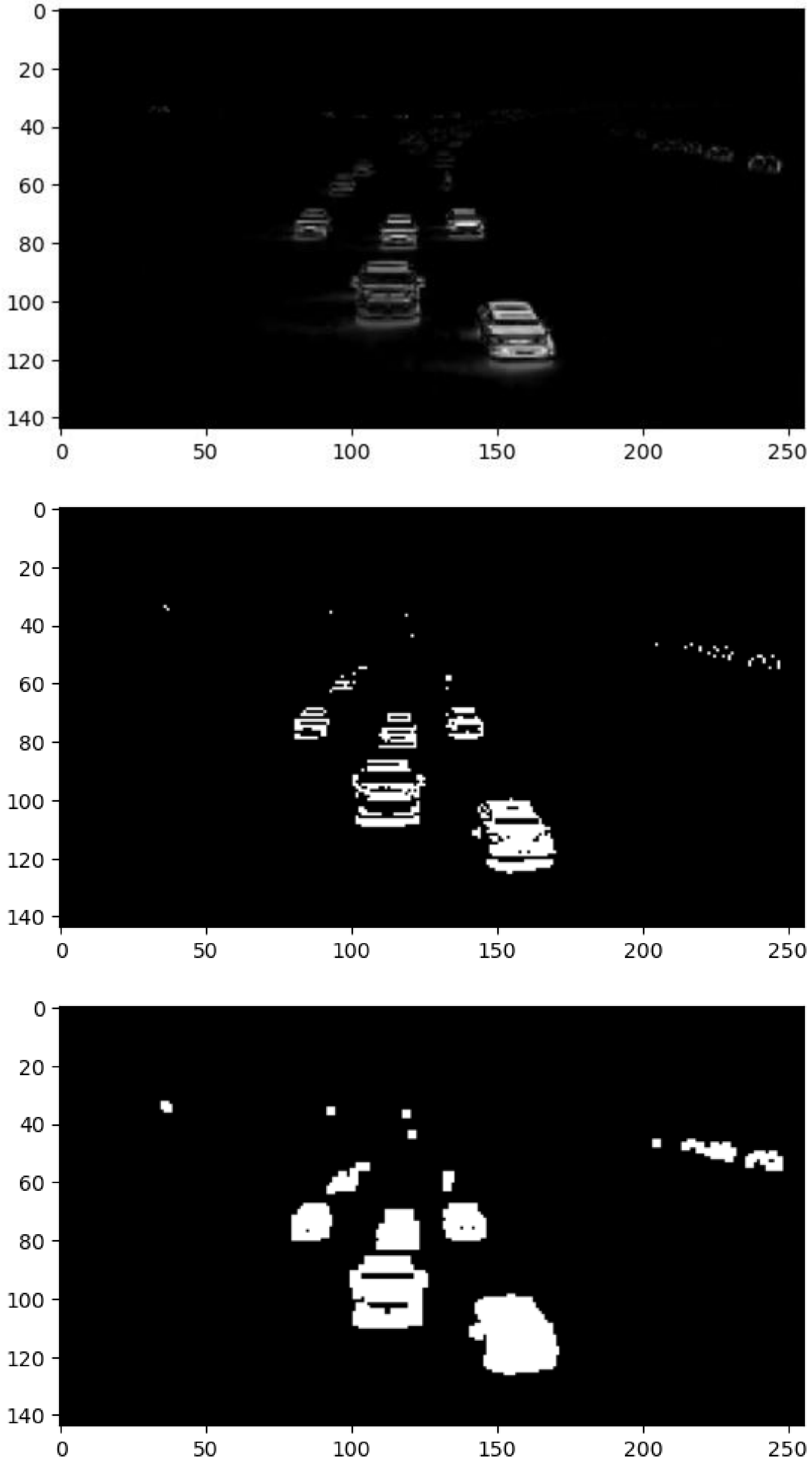
```
In [12]:
number_of_lanes = 4
no_of_vehicles = []

for i in range(number_of_lanes):
    file_name = "number_of_vehicles_lane_" + str(i+1) + ".txt"
    f = open(file_name, "r")
    no_of_vehicles.append(int(f.readline()))

baseTimer = 120 # baseTimer = int(input("Enter the base timer value"))
timeLimits = [5, 30] # timeLimits = list(map(int,input("Enter the time limits ").split()))
print("Input number of vehicles:", *no_of_vehicles)

t = [(i / sum(no_of_vehicles)) * baseTimer if timeLimits[0] < (i / sum(no_of_vehicles)) * baseTimer < timeLimits[1] else min

print("\nTime for each lane:", t)
print("Total time: ", sum(t))
```

```
Input number of vehicles: 26 48 30 56

Time for each lane: [19.5, 30, 22.5, 30]
Total time:  102.0
```

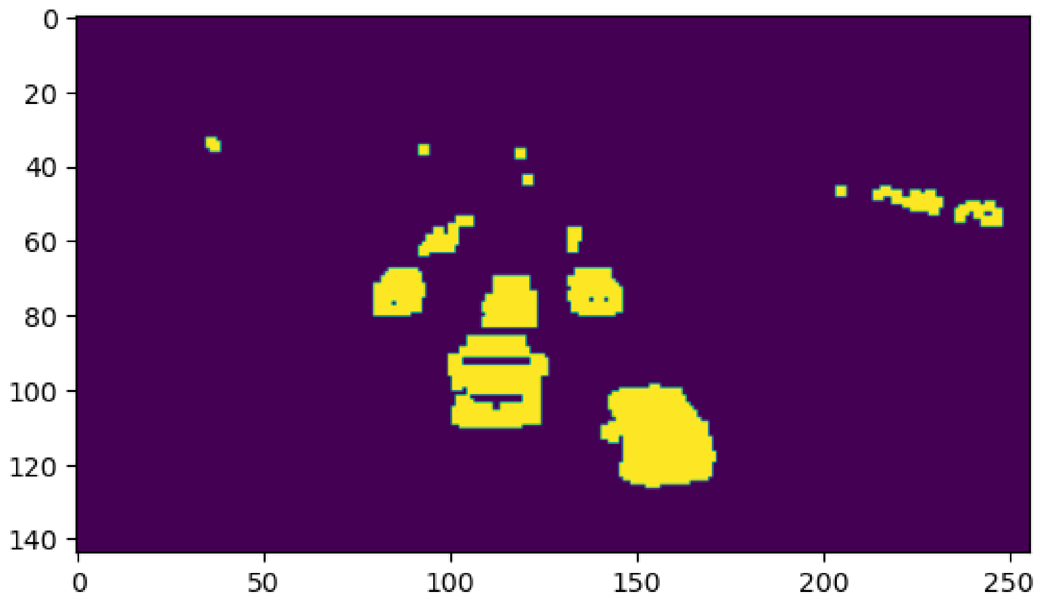
Module 2: Traffic Signal Violation Detection

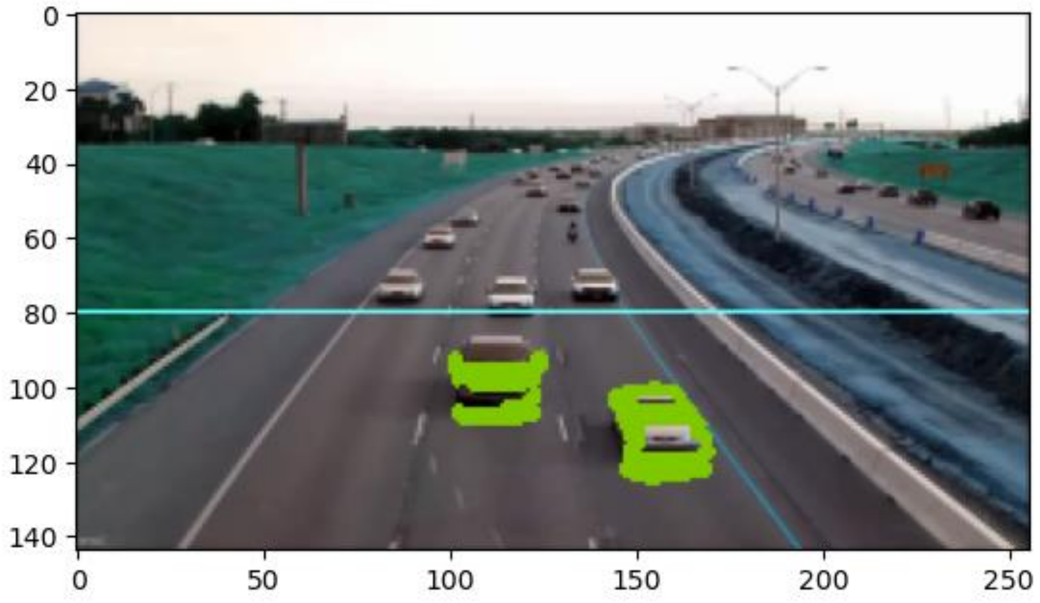
This part of our project concerns the violation of traffic signals by cars. In this procedure, we converted the video of lanes received from our cameras into individual frames. All these individual frames were compared simultaneously to each other. A line in green is also drawn in these frames and our algorithm detects any car passing this arbitrary as a traffic signal violation which is then added to our counter of violations.



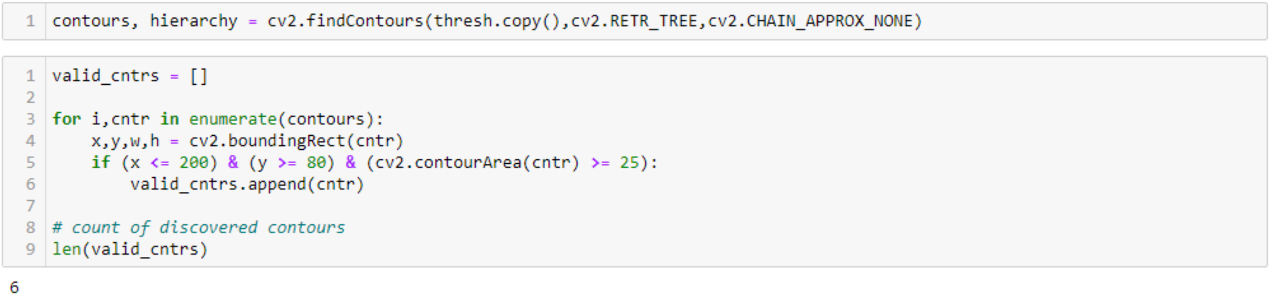


0
20
40
60
80
100
120
140
0
50
100
150
200
250
0
20
40
60
80
100
120
140
0
50
100
150
200
250
0
20
40
60
80
100
120
140
0
50
100
150
200
250





Code:



```
contours, hierarchy = cv2.findContours(thresh.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
```

```
valid_cntrs = []

for i,cntr in enumerate(contours):
    x,y,w,h = cv2.boundingRect(cntr)
    if (x <= 200) & (y >= 80) & (cv2.contourArea(cntr) >= 25):
        valid_cntrs.append(cntr)

# count of discovered contours
len(valid_cntrs)
```

```
6
```

Module 3: Speed detection using optical flow:

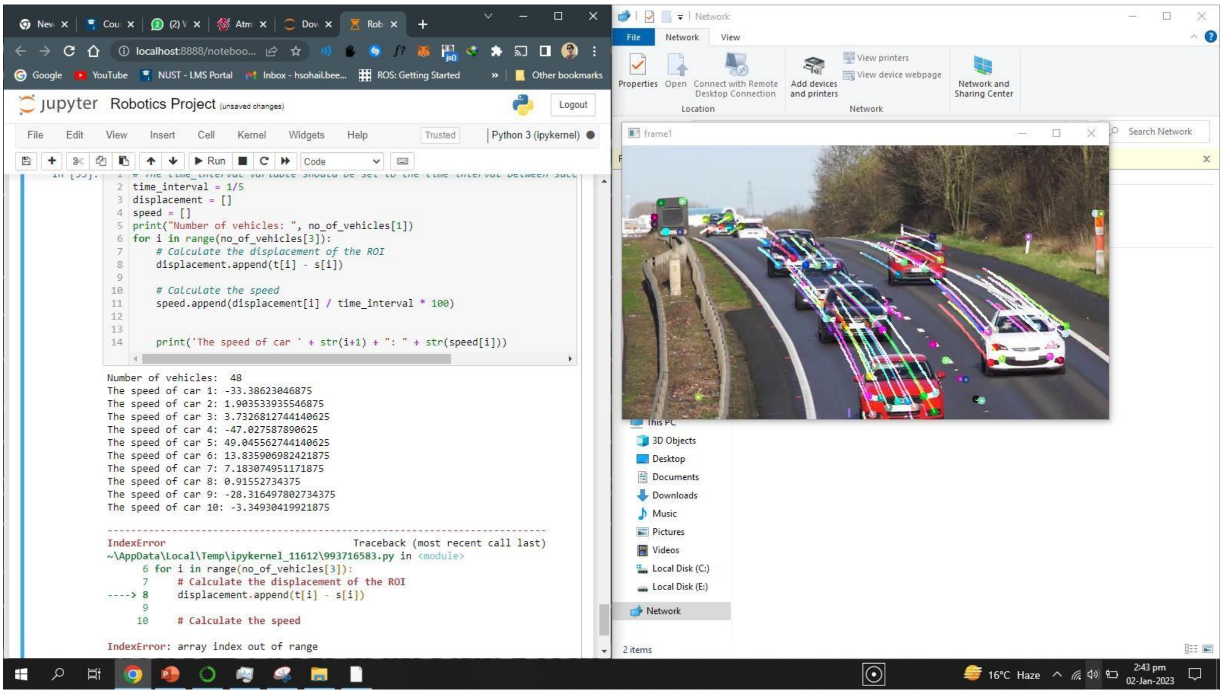
Optical flow is a method used to estimate the motion of objects in an image or video. It works by analyzing the movement of pixels between consecutive frames in a video andusing that information to calculate the speed of the objects in the scene. This can be useful for detecting the speed of vehicles because the motion of the vehicles can be tracked over time, allowing their speed to be calculated. There are several methods for calculating optical flow, including block matching, Lucas-Kanade, and Horn-Schunck. These methods can be used to accurately estimate the speed of moving objects in a scene, includingvehicles.

Optical flow is an extremely fundamental concept that is utilized in one form or another in most video-processing algorithms. Our project aims to detect the speed of the vehicles using image processing techniques and algorithms of computer vision. Such a system will helpprevent traffic violations by keeping a check of the speed of the vehicles.

Working:

Our code uses the Lucas-Kanade method to calculate the optical flow of objects in a video. The optical flow is calculated by analyzing the movement of "corners" (features with high contrast) between consecutive frames in the video. The movement of these corners is usedto estimate the motion of the objects in the scene. The code then uses this information to draw lines on the video frames to visualize the motion of the objects. The lines are drawn from the positions of the corners in the previous frame to their positions in the currentframe. The code also displays the resulting video frames with the lines overlaid on them.

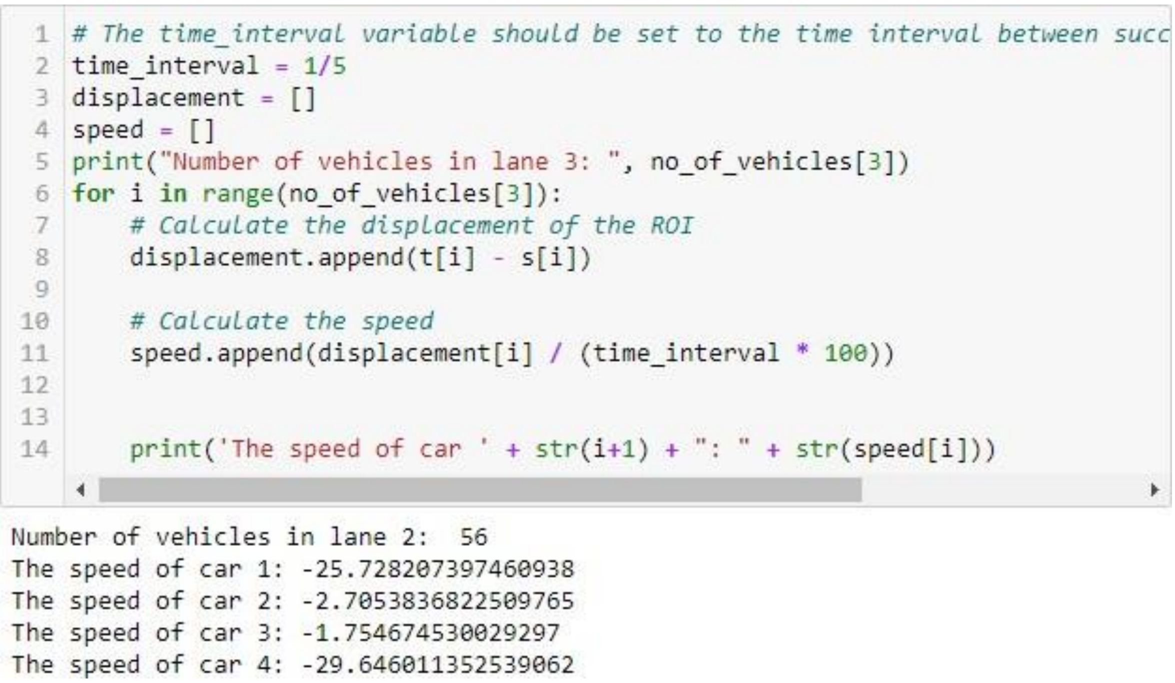
Results:



jupyter Robotics Project (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
frame1
Network

```
# The time_interval variable should be set to the time interval between succ
time_interval = 1/5
displacement = []
speed = []
print("Number of vehicles: ", no_of_vehicles[1])
for i in range(no_of_vehicles[3]):
    # Calculate the displacement of the ROI
    displacement.append(t[i] - s[i])

    # Calculate the speed
    speed.append(displacement[i] / time_interval * 100)


    print('The speed of car ' + str(i+1) + ": " + str(speed[i]))
```

```
Number of vehicles:  48
The speed of car 1: -33.38623046875
The speed of car 2: 1.903533935546875
The speed of car 3: 3.7326812744140625
The speed of car 4: -47.027587890625
The speed of car 5: 49.045562744140625
The speed of car 6: 13.835906982421875
The speed of car 7: 7.183074951171875
The speed of car 8: 0.91552734375
The speed of car 9: -28.316497802734375
The speed of car 10: -3.34930419921875

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11612\993716583.py in <module>
      6 for i in range(no_of_vehicles[3]):
      7     # Calculate the displacement of the ROI
----> 8     displacement.append(t[i] - s[i])
      9 
     10     # Calculate the speed

IndexError: array index out of range
```



```
# The time_interval variable should be set to the time interval between succ
time_interval = 1/5
displacement = []
speed = []
print("Number of vehicles in lane 3: ", no_of_vehicles[3])
for i in range(no_of_vehicles[3]):
    # Calculate the displacement of the ROI
    displacement.append(t[i] - s[i])

    # Calculate the speed
    speed.append(displacement[i] / (time_interval * 100))


    print('The speed of car ' + str(i+1) + ": " + str(speed[i]))
```

```
Number of vehicles in lane 2:   56
The speed of car 1: -25.728207397460938
The speed of car 2: -2.7053836822509765
The speed of car 3: -1.754674530029297
The speed of car 4: -29.646011352539062
```

Conclusion:

In conclusion, computer vision has the potential to significantly improve traffic management and make transportation safer and more efficient. In our project we were able to count vehicles and adjust traffic light changing time accordingly. We were also able to detect the speed of thevehicles and traffic rules violators. We used synchronization logic for traffic lights time calculation and Lucas-Kanade method for speed detection.

Lucas-Kanade method is just one of many different techniques that can be used for vehicle speed detection, and the choice of method will depend on the specific requirements and constraints of the application. Other approaches that may be suitable for this task includeradar, laser, or sonar-based sensors, as well as computer vision techniques based on feature tracking or structure from motion.

We this our algorithm is really efficient and works in most scenarios so it can be implemented at most junctions for traffic monitoring and regulation purposes.

---THE END---