Graphic Era Deemed to be University

Dehradun, Uttarakhand

A Project Report

On

Design the traffic management system using ML/AI to Reduce the congestion on roads

SUBMITTED BY:

Dhruv Punetha

B. Tech CSE(ML)

University Roll No-2014639

Gagan Soni

B. Tech CSE(ML)

University Roll No-2014654

Submitted to:

Mr. Piyush Agarwal

Department of Computer Science and Engineering

July 2021

INTRODUCTION

In recent years Indian cities are experiencing rapid increase in vehicular population, especially personal transport. This rapid increase has put a massive strain on roadways infrastructure. And given the constraints in increasing roadway width increase in vehicle result in dropping levels of service of roads. The problem with traffic control signals is typically fixed time signal control. This causes average speed to drop below 10 kmph in some cities.

Urban traffic has number of adverse effects:

1. Increase in noise and air pollution caused due to honking done waiting in traffic as well as excess fuel burned for that excess time causing useless wastage of fuel and production of pollution.
2. Cost of travel increased by a significant margin.
3. Quality of life, satisfaction, happiness is affected. Overall mood of a person is affected.
4. It effects a person's productivity and causes an increase in stress.

Health problems of police personals-

A test confirmed that 1 out of 7 traffic policemen have lung problem in Delhi and suffer from breathing problems due to air pollution. Since they spend almost 14 hours a day managing traffic on roads where air pollution is 3 to 4 times higher the safe limit this comes as no surprise. I f we apply ai to management system we will get minimal traffic intervention which can protect these officers.

Due to this many developed nations have adopted IT based traffic management systems.

Intelligent traffic management systems-

Intelligent traffic management system (ITMS) is defined as an advanced application

aims to provide innovative services related to different modes of transport and traffic management. It enables users to be better informed and to make safer, more coordinated, efficient, and smarter use of transport networks.

These are systems where centrally controlled traffic sensors, cameras and signals regulate flow of traffic through city in response to demand.

Integrating these systems in city will –

- Reduce everyday congestion by smoothing traffic flows and prioritizing traffic in response to demand in real time.
- Reduce pollution throughout city as stop-start driving is inefficient and polluting. Much time would be saved
- We can give priority to medical and emergency services so they can reach their destination faster by detecting such vehicles.
- Avoid roadblocks and prevent accidents from happening by effectively analysing vehicles every now and then.
- Reduce waiting period for the vehicles and people and save fuel.

Interest in ITMS is growing rapidly due to increasing population density. In developing countries such as ours, migration from rural to urbanised habitats due to rapid urbanisation and industrialisation is causing high population density without significant infrastructural development of the suburbs. Mega cities, such as New Delhi and Mumbai, are affected the most. Use of multimodal transportation systems, including bicycles, motorcycles, auto rickshaws, cars, buses, metro, trains, and pedestrians, are leading to rapid increase in road traffic.

The solution to these problems is by utilizing advanced technologies and intelligent solutions by deploying ITMS. This system would be able to track the flow and pace of traffic to provide real-time traffic management, which is more dynamic and accommodative of the varying nature of traffic density.

Our Approach to Solve this problem

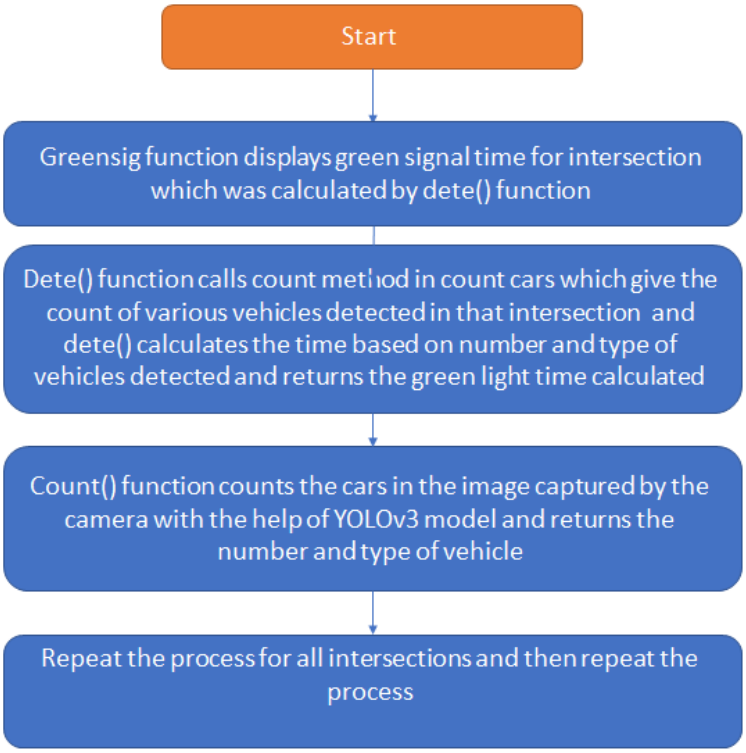
Most of the traffic congestion problems is basically due to predetermined distribution of green light time at an intersection by traffic lights and they cannot adapt to real time traffic density. The TOD signal operates on a pre-set signal cycling which cycles based on the average number of average passenger cars in the memory device of an electric signal unit. This will not work if there are more than average cars in one lane and less on the other.

Hence, we derived a real time approach for traffic light system. This model gives green light time based on number of vehicles in a lane. If there are more vehicles on one lane the green light will be on for longer duration. And if there are less cars then it would be on for lesser duration.

Not just the number of vehicles even the type would matter as not all vehicles stake same time to cross. Based on vehicle type larger vehicle will be given more time to cross.

Steps-

- Our first step is to install cameras for each lane in an intersection.
- Then we set a default time after which the light will change, and new green light time is displayed.
- Before changing the lane, an image is taken of that lane for which green light duration is to be predicted.
- This image is passed to our model which detects the number of vehicle and type of vehicle in that lane.
- The detected vehicles and their type are sent to another function which calculates the green light display duration from the number and type of vehicles.
- Green light is set for duration calculated.
- At the end of the green light timer, we go back to step 3 and repeat the process again.



To detect the number of vehicles we use pre-trained model YOLO (specifically yolov3) to perform the task of object detection. The pretrained model YOLO uses OpenCV for object detection along with multiple foreground and background subtraction and removal of noise from the input image. YOLO (You Only Look Once), is a network for object detection. It is the one of the most powerful pretrained model to give utmost accuracy and is a combined version of RCNN and SSD, both make Yolo much faster, efficient, and powerful algorithm. By applying object detection algorithm in Yolo, one will not only be able to determine what is in an image, but also where a given object is placed.

After counting vehicles with the help of yolo the result is passed to a function that calculates display time. In this function we have assigned each type of vehicle specific weights (average time that they would take to cross) and according to this we calculate the total time it would take the vehicles to cross.

HELP Taken From-

1 Articles that helped understand the issue of traffic management and various problems it causes-

- https://scroll.in/article/876209/traffic-jams-in-indian-cities-arent-just-frustrating-theyre-also-expensive
- https://repositorio.cepal.org/bitstream/handle/11362/37898/1/LCG2199P_en.pdf
- https://traveltips.usatoday.com/effects-traffic-congestion-61043.html
- https://www.geotab.com/blog/traffic-congestion/
- https://bklyner.com/what-really-causes-traffic-congestion-sheepshead-bay/

2 Articles that helped me understand how machine learning can help solve this issue-

- https://ieeexplore.ieee.org/abstract/document/9077777
- https://www.researchgate.net/publication/330858371_Adaptive_Traffic_Management_System_Using_IoT_and_Machine_Learning
- https://www.researchgate.net/publication/342238898_Traffic_Signal_Management_using_Machine_Learning_Algorithm
- https://techwireasia.com/2020/08/ai-powered-traffic-management-is-slashing-asias-congestion-problem/
- https://ieeexplore.ieee.org/document/8482380

3 Articles related to use of YOLO and it's understanding its working-

- https://www.mdpi.com/2076-3417/10/9/3079/pdf
- https://pjreddie.com/darknet/yolo/
- https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006
- https://youtu.be/h56M5iUVgGs
- https://appsilon.com/object-detection-yolo-algorithm/

4 GitHub repositories that helped-

- https://github.com/VinayNR/Traffic-Control-System-using-Deep-Q-Learning
- https://github.com/wingsweihua/IntelliLight
- https://github.com/suessmann/intelligent_traffic_lights
- https://github.com/Twitwi96/Smart-traffic-light-2
- https://github.com/Baazigar007/Smart-Traffic-Light
- https://github.com/jideilori/vehicle-counting
- https://github.com/guptavasu1213/Yolo-Vehicle-Counter

OUR WORK

Various code we worked on and understood to develop this model.

1 Understanding OpenCV and how to capture frames with its help so that it can be used to capture pictures at the intersection. Link to code- https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/blob/main/SmallerCodes/imagecapture.py

23 lines (23 sloc) | 702 Bytes

```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
if not cap.isOpened(): #if camera cant be opened
    print("Cannot open camera")
    exit()
while True:
    # Capture frame-by-frame
    ret, frame = cap.read() #returns bool value
    print(ret)
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    # Our operations on the frame come here
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # Display the resulting frame
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv.destroyAllWindows()
```

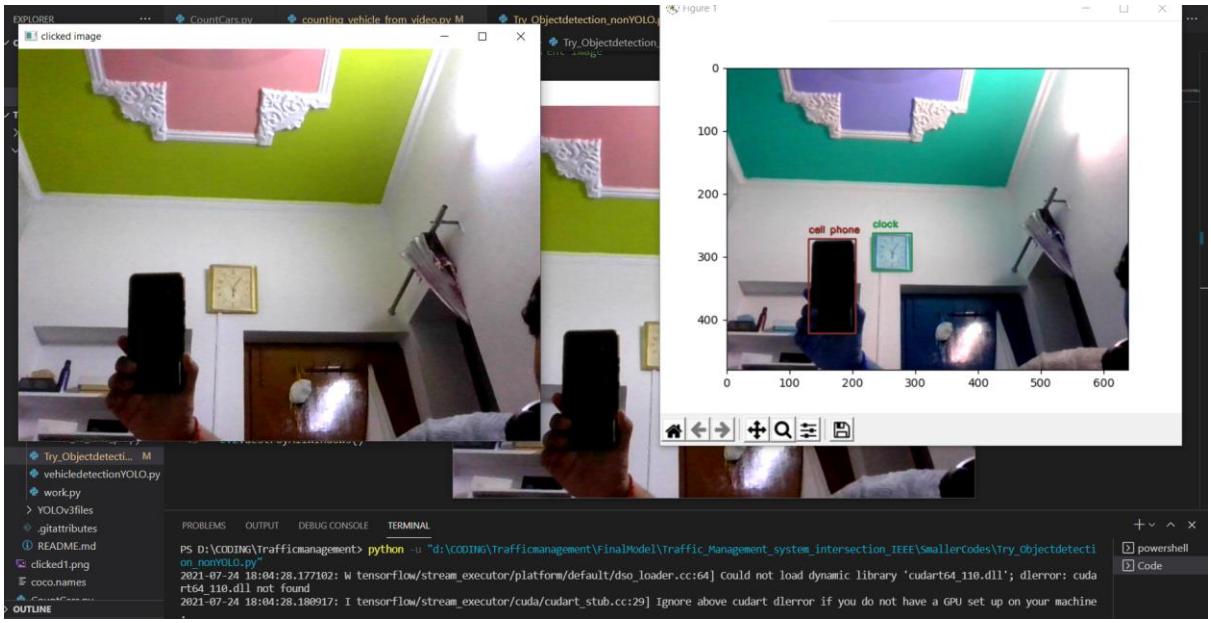
2 Saving video/photo with the help of web cam. (As the image taken by the cam is passed to model to count cars). Link to code- https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/blob/main/SmallerCodes/saving_video_from_web_cam.py

```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
# Define the codec and create VideoWriter object
fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('mya.avi', fourcc, 20.0, (640,  480))
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    frame = cv.flip(frame, 0)
    # write the flipped frame
    out.write(frame)
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break
# Release everything if job is finished
cap.release()
out.release()
cv.destroyAllWindows()
```

3 Object detection using OpenCV (NON-YOLO)- Worked on this code to have see how object detection works and understand it's working.

Link to code-
https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/blob/main/SmallerCodes/Try_Objectdetection_nonYOLO.py

```python
import cv2
import time
import matplotlib.pyplot as plt
import cvlib as cv
from cvlib.object_detection import draw_bbox
cp=cv2.VideoCapture(0) #capturing the video using camera to take pictures
imnum=10 #image number that we are about to take
while True:
    ret, img = cp.read()
    cv2.imshow('image show', img)
    k = cv2.waitKey(1) #wait key specifies the wait time before next frame
    # set the key for the countdown
    Timer = int(2)  #our timer in hich we take the next image
    if k == ord('q'):
        prev = time.time()
        while Timer >= 0:
            ret, img = cp.read()
            font = cv2.FONT_ITALIC
            cv2.putText(img, str(Timer),(50, 50), font,2, (0, 144, 170))
            cv2.imshow('image show', img) #show current image
            cv2.waitKey(1)
            cur = time.time() #current time
            if (cur-prev >= 1):
                prev = cur
                Timer = Timer-1

        ret, img = cp.read()
        cv2.imshow('clicked image', img)
        cv2.waitKey(2000)
        cv2.imwrite('Traffic_Management_system_intersection_IEEE\SmallerCodes\clicked'+str(imnum)+'.png', img)
        im = cv2.imread('Traffic_Management_system_intersection_IEEE\SmallerCodes\clicked'+str(imnum)+'.png')
        imnum=imnum+1

        bbox, label, conf = cv.detect_common_objects(im)
        output_image = draw_bbox(im, bbox, label, conf)
        plt.imshow(output_image)
        plt.show()
        #print('Number of cars in the image is '+ str(label.count('car')))
    elif(k==27):
        break
```

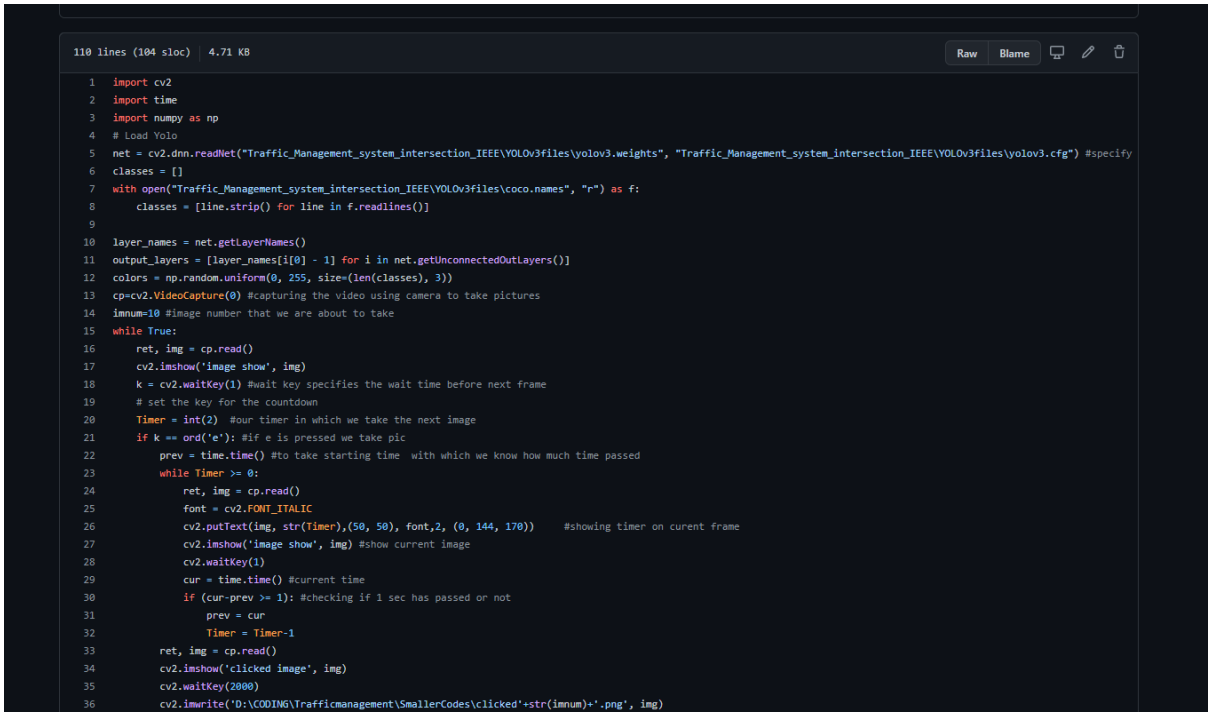


4 Vehicle detection with the help of YOLO- Here we used YOLO (yolov3) to count number and type of vehicle from the clicked image.

Link to code-

https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/blob/main/SmallerCodes/vehicledetectionYOLO.py

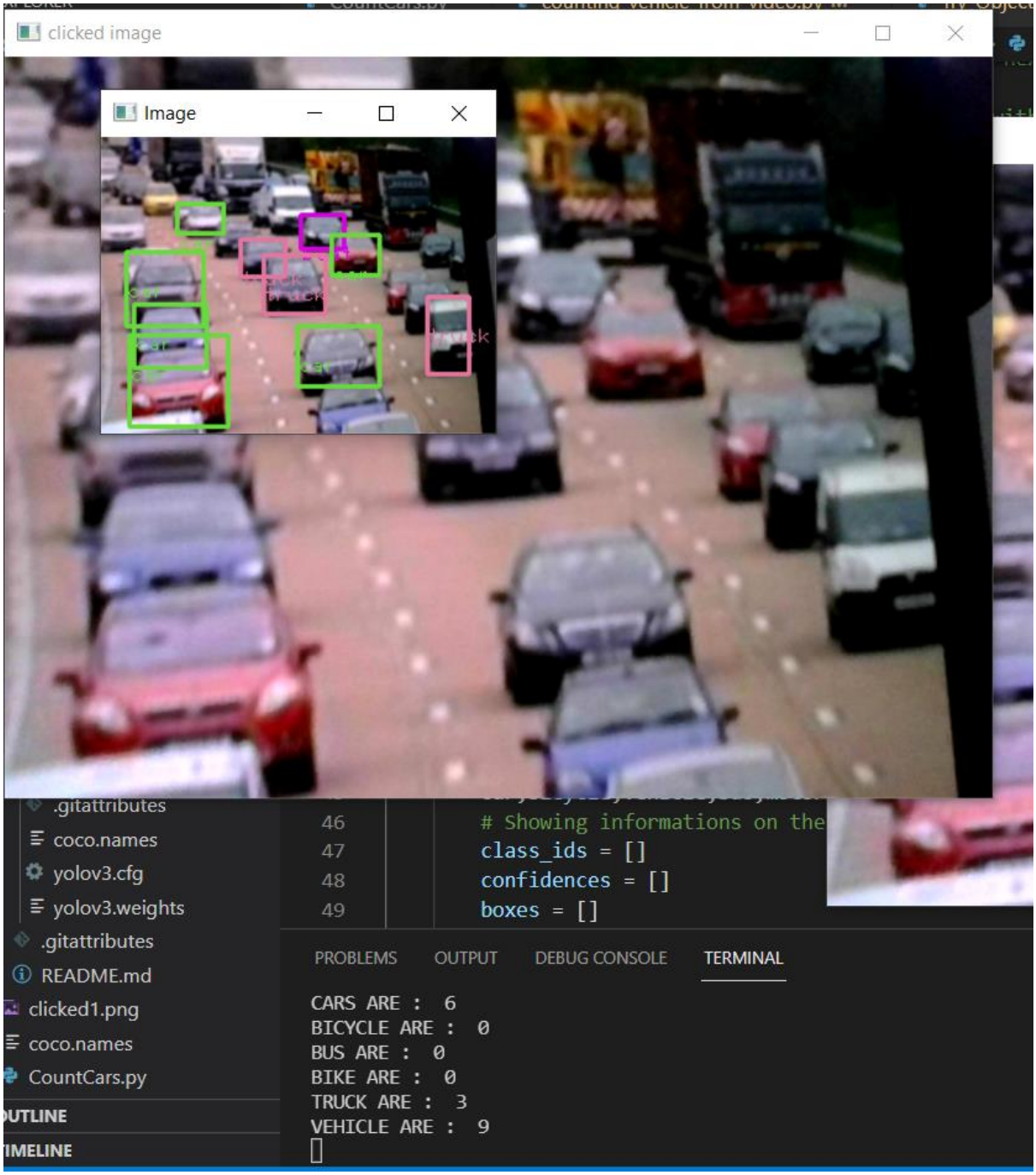
https://github.com/iamgagansoni/Traffic-Management/blob/main/Car_Detection/car_detection.py



110 lines (104 sloc) | 4.71 KB

Raw Blame

```python
import cv2
import time
import numpy as np
# Load Yolo
net = cv2.dnn.readNet("Traffic_Management_system_intersection_IEEE\YOLOv3files\yolov3.weights", "Traffic_Management_system_intersection_IEEE\YOLOv3files\yolov3.cfg") #specify
classes = []
with open("Traffic_Management_system_intersection_IEEE\YOLOv3files\coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
cp=cv2.VideoCapture(0) #capturing the video using camera to take pictures
imnum=10 #image number that we are about to take
while True:
    ret, img = cp.read()
    cv2.imshow('image show', img)
    k = cv2.waitKey(1) #wait key specifies the wait time before next frame
    # set the key for the countdown
    Timer = int(2)  #our timer in which we take the next image
    if k == ord('e'): #if e is pressed we take pic
        prev = time.time() #to take starting time  with which we know how much time passed
        while Timer >= 0:
            ret, img = cp.read()
            font = cv2.FONT_ITALIC
            cv2.putText(img, str(Timer),(50, 50), font,2, (0, 144, 170))     #showing timer on curent frame
            cv2.imshow('image show', img) #show current image
            cv2.waitKey(1)
            cur = time.time() #current time
            if (cur-prev >= 1): #checking if 1 sec has passed or not
                prev = cur
                Timer = Timer-1
        ret, img = cp.read()
        cv2.imshow('clicked image', img)
        cv2.waitKey(2000)
        cv2.imwrite('D:\CODING\Trafficmanagement\SmallerCodes\clicked'+str(imnum)+'.png', img)
```



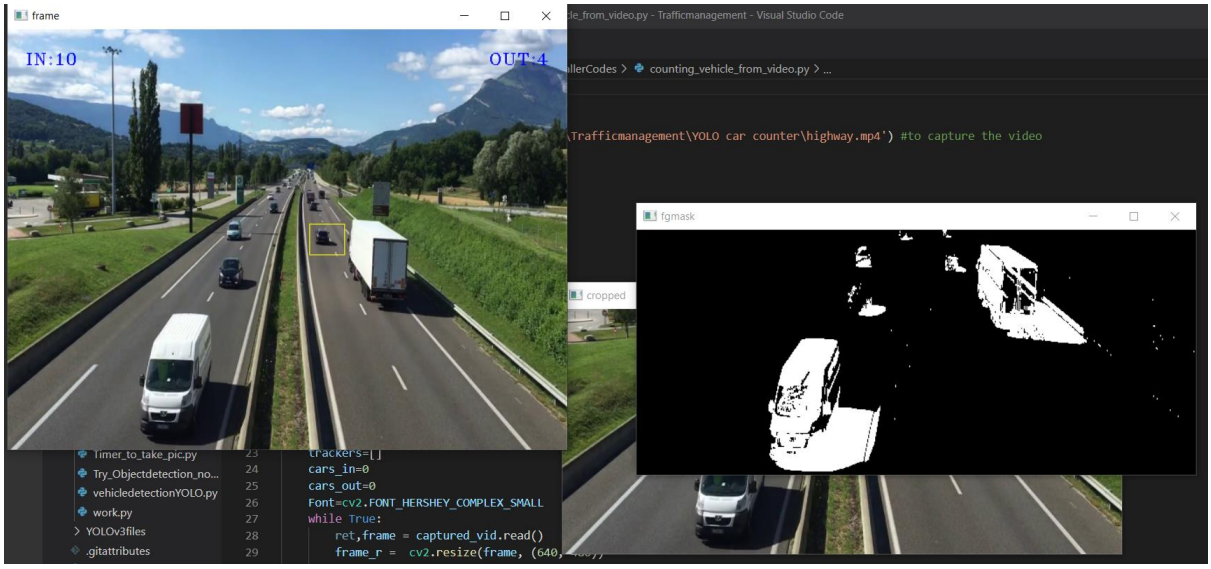
5 Background subtraction method to count vehicles in a video- we were exploring various methods through which vehicle could be counted this method was faster than yolo but was not as accurate as it was counting vehicles based on movement and hence could count other entities as vehicles.

Also, it was impossible to classify the type of vehicle with this method as it was using background subtraction.

114 lines (94 sloc) | 3.95 KB

Raw Blame

```python
import cv2
import dlib

captured_vid = cv2.VideoCapture('Traffic_Management_system_intersection_IEEE\SmallerCodes\highway.mp4') #to capture the video

def findCenter(x,y,w,h):
    cx = int((x+w)/2)
    cy = int((y+h)/2)
    return cx,cy

def pointInRect(x,y,w,h,cx,cy):
    x1, y1 = cx,cy
    if (x < x1 and x1 < x+w):
        if (y < y1 and y1 < y+h):
            return True
    else:
        return False

def main():
    fgbg =  cv2.bgsegm.createBackgroundSubtractorMOG()
    count=0
    SKIP_FRAMES=10
    trackers=[]
    cars_in=0
    cars_out=0
    Font=cv2.FONT_HERSHEY_COMPLEX_SMALL
    while True:
        ret,frame = captured_vid.read()
        frame_r =  cv2.resize(frame, (640, 480))
```

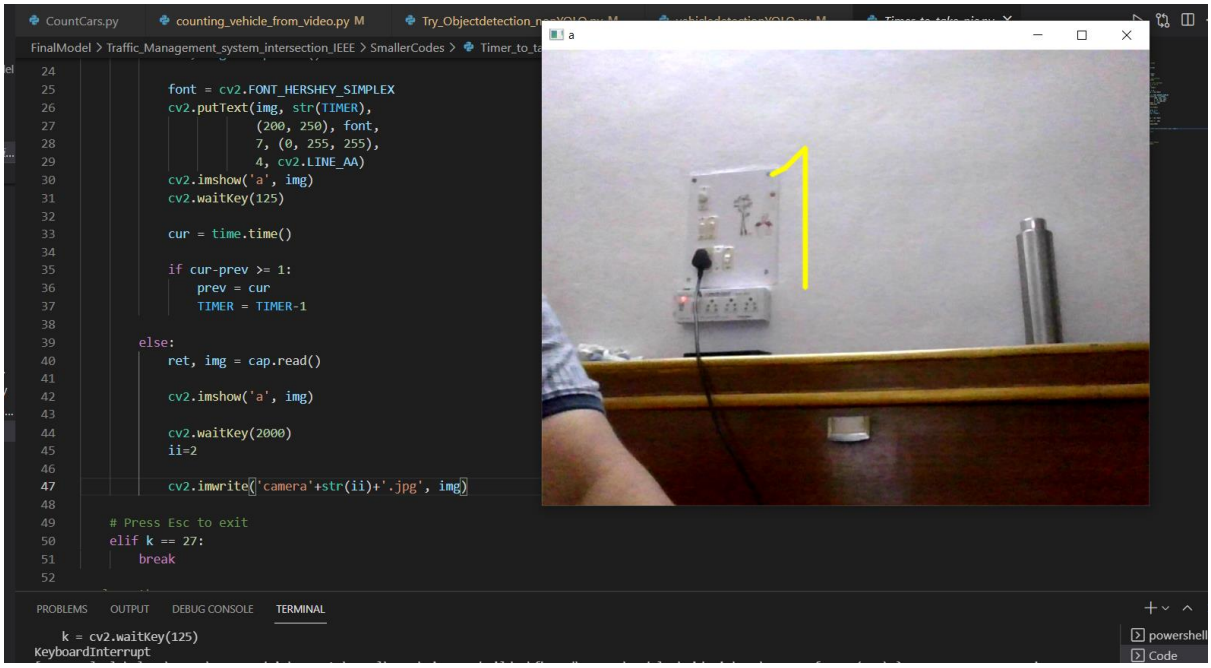


6 Timer understanding- How to implement timer that will be used to control the cam taking pictures.

Link to code-

https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/blob/main/SmallerCodes/Timer_to_take_pic.py

```python
import cv2
import time
# SET THE COUNTDOWN TIMER
TIMER = int(5)
# Open the camera
cap = cv2.VideoCapture(0)
while True:


    ret, img = cap.read()
    cv2.imshow('a', img)

    # check for the key pressed
    k = cv2.waitKey(125)

    # set the key for the countdown
    # to begin. Here we set q
    if k == ord('q'):
        prev = time.time()

        while TIMER >= 0:
            ret, img = cap.read()

            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img, str(TIMER),
                        (200, 250), font,
                        7, (0, 255, 255),
                        4, cv2.LINE_AA)
            cv2.imshow('a', img)
            cv2.waitKey(125)

            cur = time.time()

            if cur-prev >= 1:
                prev = cur
                TIMER = TIMER-1

        else:
            ret, img = cap.read()

            cv2.imshow('a', img)

            cv2.waitKey(2000)
            ii=2
```



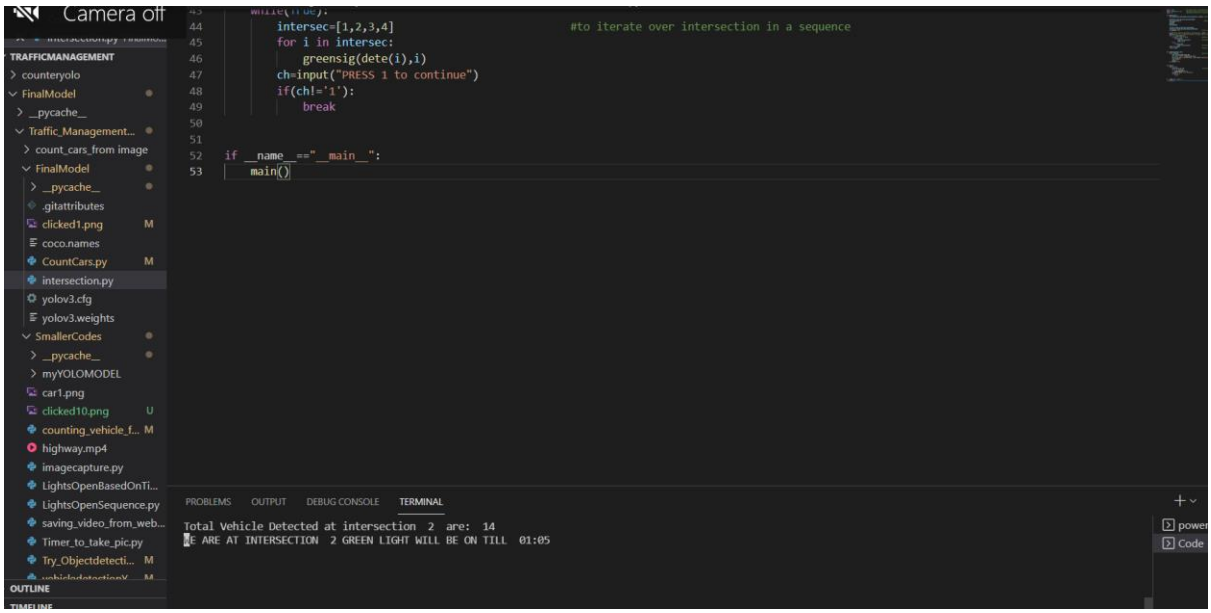
7 Final Model- this is the final model that detects cars in image and their type and based on that assign green light time for that lane and after the timer ends it takes image for other lane and calculates the time.

Link to code-

https://github.com/dhruvpunetha/Traffic_Management_system_intersection_IEEE/tree/main/FinalModel

https://github.com/iamgagansoni/Traffic-Management/blob/main/trafficlight.py

```python
import cv2          #for using computer vision to take images
import numpy as np  #numerical python library to work with various datastructures and arrays

                    # count is a function that is counting cars from a image

def count():

    # Loading Yolo

    net = cv2.dnn.readNet("Traffic_Management_system_intersection_IEEE\FinalModel\yolov3.weights", "Traffic_Management_system_intersection_IEEE\FinalModel\yolov3.cfg") #specif
    classes = []                                                    #for stroing various things yolo can detect

    with open("Traffic_Management_system_intersection_IEEE\FinalModel\coco.names", "r") as namef: #extracting name of various oject yolo can detect from its coco.name file
        classes = [line.strip() for line in namef.readlines()]      #classes is storing all the various classes yolo can detect

    layer_names = net.getLayerNames()                   #It gives you list of all layers used in a network. yolo v3 has 250

    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

    cp=cv2.VideoCapture(0)                              #capturing the video using camera to take pictures

    imnum=1                                             #image number that we are about to take we are storing the number of images


    ret, img = cp.read()                                #"Frame" will get the next frame in the camera while "Ret" will obtain return value from getting the camera frame, ei

    cv2.imwrite('Traffic_Management_system_intersection_IEEE\FinalModel\clicked'+str(imnum)+'.png', img) #storing this image

    img = cv2.imread('Traffic_Management_system_intersection_IEEE\FinalModel\clicked'+str(imnum)+'.png') #using this image
    imnum=imnum+1                                       #incrementing this so next image will be called clicked n+1

    img = cv2.resize(img, None, fx=0.4, fy=0.4)         #resizing image as per our use
    height, width, channels = img.shape
    # Detecting objects

    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False) #returns a blob converted by yolo
    net.setInput(blob)                                  #Sets the new input value for the network.

    outs = net.forward(output_layers)                   #compute output of layer with name

    car,bicycle,vehicle,bus,motorbike,truck=0,0,0,0,0,0   #keeping counts of various vehicles
    class_ids = []
```



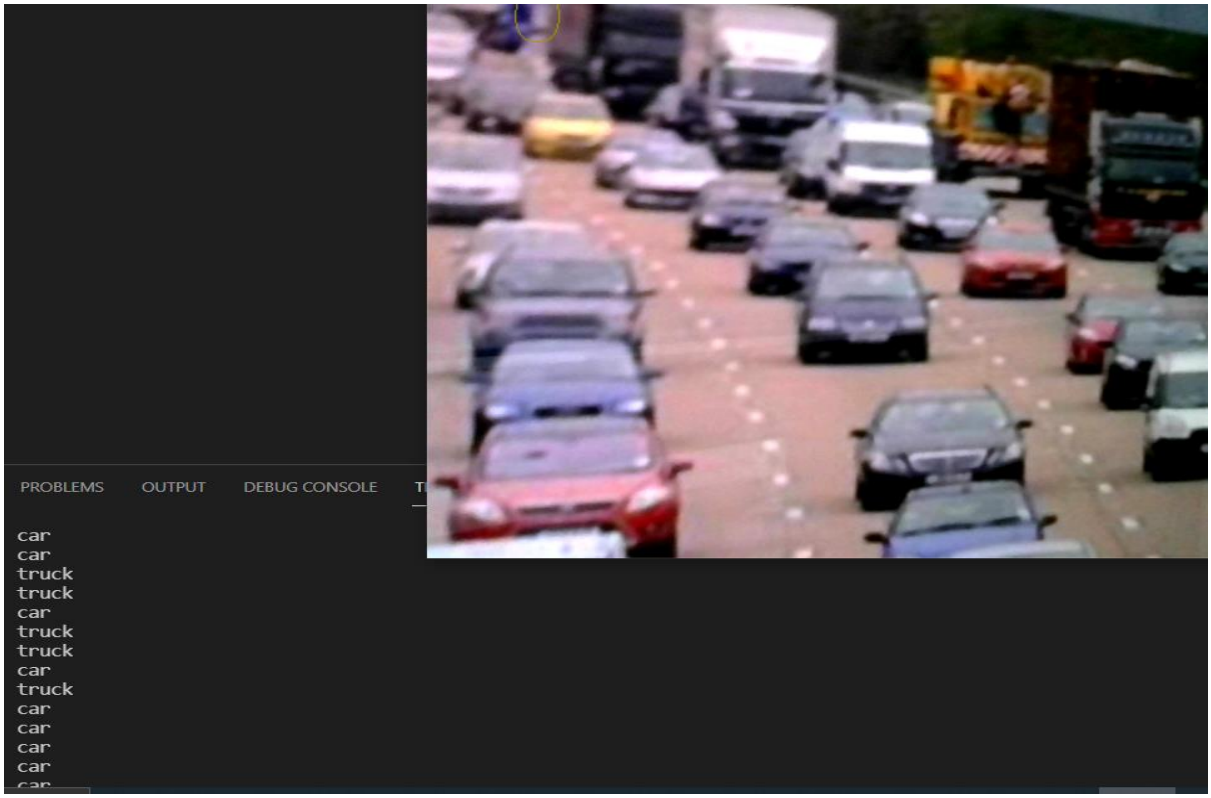
IMPROVEMENTS

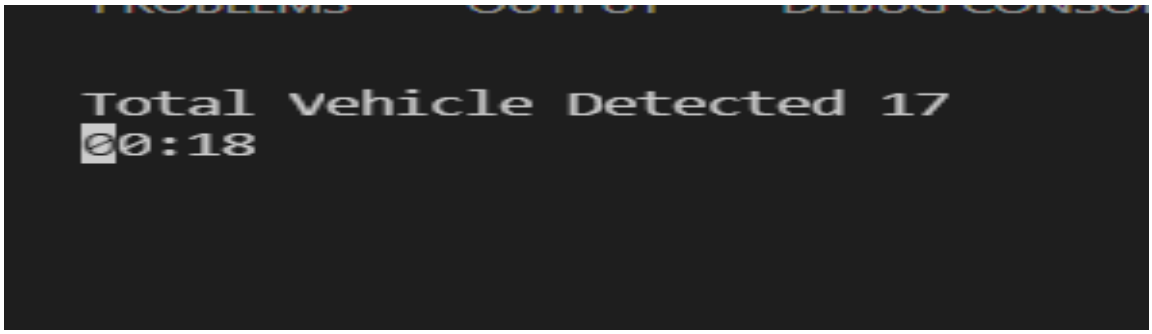
Nothing can ever be perfect there is always scope for improvement. Just like that there are various improvements that can be done in this project-

- For instance, instead of using weighted if else conditions we can train another algorithm for calculating time that is more accurate.
- We can further classify cars to identify emergency services such as ambulance and police automobile. Which can be given more priority.
- Instead of using camera we can also use sensors to detect the cars which are more accurate.
- Yolo v4 can be used instead of yolov3 as it is the latest version of the model.
- Instead of clicking picture and counting vehicle in that we can do live prediction for the video this is more accurate.

RESULTS

We were able to develop a model that can do better traffic management at an intersection as compared to the traditional approach.




Total Vehicle Detected 17
00:18