VEHICLE DETECTION AND TRAFFIC FLOW TRACKING

BY

SURYA SHANKAR BRAHMA
ID: 162-15-8019

MD. HASANUZZMAN
ID: 162-15-8129

SUMAYA HAQUE
ID: 162-15-8048

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Mr. AHMED AL MAROUF
Lecturer
Department of CSE
Daffodil International University

Co-Supervised By

MR. SHAH MD. TANVIR SIDDIQUEE
Assistant Professor
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

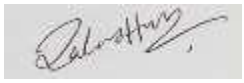
October 2020

APPROVAL

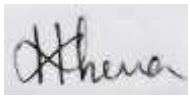
This Project/internship titled **"Vehicle Detection and Traffic Flow Tracking"**, submitted by Surya Shankar Brahma, ID No: 162-15-8019 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 08 October , 2020.

BOARD OF EXAMINERS

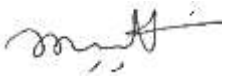
Dr. Syed Akhter Hossain — **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Md. Zahid Hasan — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Most. Hasna Hena **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Dr. Mohammad Shorif Uddin **External Examiner**
Professor
Department of Computer Science and Engineering
Jahangirnagar Univers

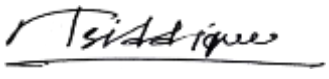
DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Mr. Ahmed Al Marouf , Lecturer, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

Mr. Ahmed Al Marouf
Lecturer
Department of CSE
Daffodil International University

Co- Supervised by:



Mr. Shah Md. Tanvir Sidiquee
Assistant Professor
Department of CSE
Daffodil International University

Submitted by:

Surya Shankar Brahma
ID: -162-15-8019
Department of CSE
Daffodil International University

Sumaya Haque
ID: -162-15-8048
Department of CSE
Daffodil International University

Md. Hasanuzzaman
ID: -162-15-8129
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Mr. Ahmed Al Marouf,** Lecturer, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of Augmented Reality to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Sayed Akhter Hossain, Professor and Head**,** Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

The purpose behind this project is to detect, track & count vehicles from a real time camera & analyze if there is any traffic jam or not. In our country, specifically in Dhaka, we are suffering from severe traffic jam problem as the number of vehicles have become exponentially large & shortage of roads. Traffic drains our valuable time. If we could have a Real-time traffic update, the authority could have rescheduled the signals accordingly. In order to minimalize this issue, we are trying to develop a system that will address the problems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Introduction

One of the biggest problems in the modern era is traffic problem. Especially for populated countries like Bangladesh as the number of vehicles have become exponentially large & shortage of roads in every city. People of each sectors can relate with this issue. In everyday problems, this is total time consuming of all. There are many reasons behind this crux like, unplanned road initiatives, occupation of footpath by hawkers, vehicle outnumber, narrow roads etc. As a result, students, employees, medical staff and every other person is facing this suffering. We cannot shorten the population but we certainly can reduce the everyday suffering. In these modern days, we are witnessing a lot of changes in our day to day life which were seem to be impossible few decades ago. What if we could have a Real-time traffic update, the authority could have rescheduled the signals accordingly.

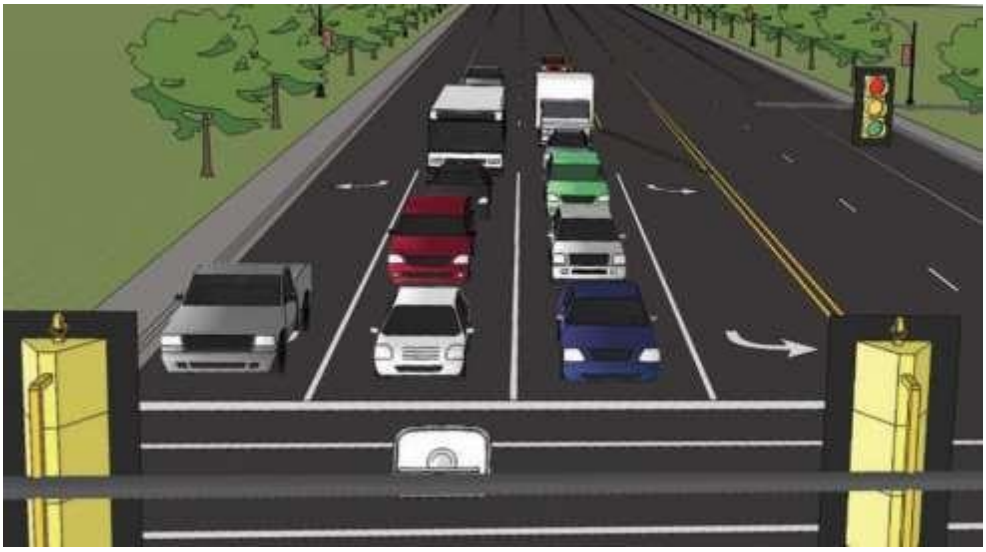


Fig 1.1: An organized picture of vehicle management.

1.2 Motivation

We have focused to develop a system that can detect real-time vehicle movement. It basically starts with traffic surveillance (video based). Researchers have been working on modern technologies to improve the whole transport system for years including transport planning, management and surveillance to make them more useful for us. This project idea is come from those thoughts that makes our life easier where we do not have to suffer much while going out.

Objectives:

- Real-time traffic update.
- Predicting suitable traffic situation.
- Analyze traffic update.
- Vehicle count.
- Assuring before going out.

1.3 Rationale of the study

We need to compute the tracking system first. In our project, we have built a software with a purpose to count the number of vehicles on road by using “OpenCV” and “Python”.

First, we shall count the number of vehicles that are heading upwards or downwards of our line of interest which is located at the center of the screen from a live or offline video footage. But first we must make sure our knowledge about what is object detection and object tracking. Then we accomplish the right way of vehicle tracking and count. It will:

- set a sole number/ID to the specific object.

- When it moves, track its movement and predict new location of the object depends on many frame attributes.

Object tracking algorithms has many examples like MedianFlow, Kernalized correlation filter etc.

1.4 Research questions

Research Questions part is very important for any research project. It is important to be focused, described and has a point. Basically, the research questions are relevant to the research project and need to be answered directly through the analysis of the data. The research questions relevant to our project that we will be trying throughout the research paper are given below:

- What kind of data's we are using to complete our project analysis?
- How to prepare the dataset?
- The right algorithm behind the analysis?
- How to compare this method with other methods in the corresponding fields?
- Who will be the user for the project?
- Why use the project?

1.5 Research outcome

Our research is expected to provide the real-life solution of problems. Some of them are:

- A full featured computer vision-based product to detect and track the vehicles in real time.
- An Interactive real-time interface to visualize the tracking system.
- A business model to engage the product in human loops.
- Define the prediction by the result.

1.6 Layout of report

Our project report is organized as follows:

- First covers our project introduction, project motivation, research questions, expected outcome and layout of our report.
- Chapter two covers literature review, our point of view, related work, project research summary, scope the problems and challenges.
- Third Chapter describes our research methodology.
- Chapter four covers codes implementation and the experimental results.
- Chapter five includes our project's future work and conclusion.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Now a days, we can do any task with the help of modern technology. This is what made our life more reliable and comfortable. Internet is a big and powerful thing among them. No one can imagine this modern world without the internet. There are various types of facilities that are popular for their online services, but it will be very effective and useful shopping if people can use this popular medium for something informative. Something that can assure their proper time maintenance. If we could have a Real-time traffic update through the internet, the controller could have rescheduled the signals accordingly to the traffic. To establish the purpose, we would need to detect the vehicles, count the vehicle numbers and track their movement and show the overall progress. Which is why we are trying to develop a system that will address the problems.

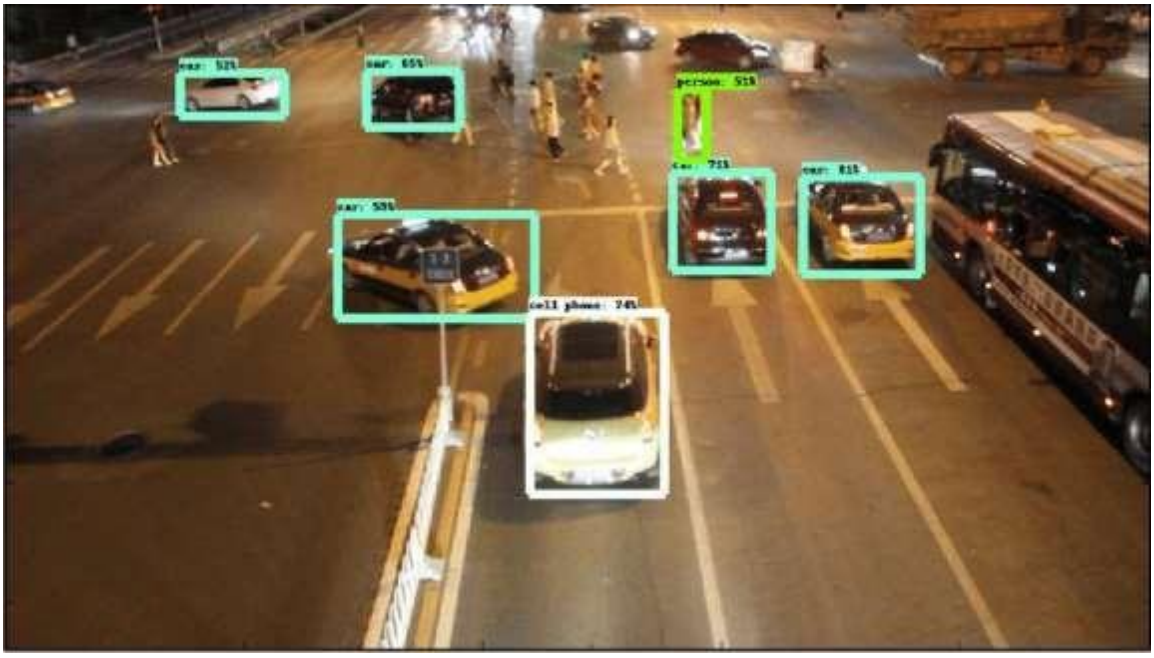


Fig 2.1: A prototype of Vehicle Detection in Retail.

2.2 Related work

Bluetooth based Automatic Vehicle Detector

The project is the result of AVI data collection. It has a physical set-up of the data collections and the interpretation of the system. Statistical before-after examples tests are also provided here. Basically, the systems uses Bluetooth address matching (MAC). Although the concepts come from different AVI data sources.

An Intelligent Vehicle Counting System for Traffic Management

This project is designed to analyze counting next to a very high accuracy. Even is difficult scenarios like dark shadow or in night time, the system is programmed to get a great accuracy number. Its based on real-time vehicle detection by analyzing video footages and its accuracy it 98%.

Au system to localize and recognize vehicle plate numbers

The research addresses a system that capture a plate image by a camera and summarizes the characteristics of the plate. After then, by using a multilayer neural network it identifies symbols from the number plate.

On-Road Vehicle Detection

The research presents an overview of recent vision based vehicle detection system on road. It is a system that designed to mount on the vehicle instead of fixing a traffic monitoring system.

2.3 Summary of our research

Detection of vehicles and tracking them in the roads and managing the traffic is a remarkable gift as vehicle management itself. By installing traffic cameras, a colossal database has been analyzed regarding traffic management. It would be better if the camera is located on a high ground for a better view because object sizes changes comprehensively in this angle. We focused on a reliable solution based on the issue. After getting a result, we display it in command prompt as a screen where it shows vehicle counting, detecting and tracking.

2.4 Scope the problem

In this scenario

- Analyze different types of camera angles.
- The data's format changing problem.
- Accuracy maintenance.
- Object motion speed track.

2.5 Challenges

Challenges that we face

- Camera motion is a big problem.
- Detection of different Shapes of Vehicles.
- Multiscale Problem.
- Dynamic Changes in the movement

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter covers the informations that carries various theories and concepts. Initially we used High Level Programming Language Python and "PyCharm" as programming platform although it can be accomplished in different other language and platforms. We chose this platform because Python is more productive than any other programming languages and it handles a lot of complexity. Also, another reason for using Python is it allows you not to waste time with confusing syntax. So far it is best suitable platform where we can perform this experiment.

Fig 3.1: Working platform of python with PyCharm

First of all, we will discuss about the necessary Python packages that is required to create vehicle counter.

Then, we will have a deliberation on combining counter, object tracking and object detecting stages.

Eventually, we will enquire into the results of counting with OpenCV to video footages.

3.2 Tools & necessary libraries for project

Given files will be used for this program:

Numpy – It is used for easier matrix operations.

OpenCV- It is used for opening and waiting video files, pre-train deep neural network and displaying output screen/frame.

Imutils – It is used to add more assisting function to OpenCV.

Dlib – It is used for implementing its correlation of tracker algorithm.

3.3 Difference between object tracking and object detecting

Before moving on to the main point, let's understand the difference between object tracking and object detection.

Where an object is placed in an image or frame is called Object detection. An object detector is also expensive in terms of computation and very much slower than object tracking algorithm. There are lots of examples of object tracking algorithm like HOG+ Linear SVM, Haar casacades, also object detectors like YOLO, SSDs and Faster R-CNNs.

On the other hand, An object tracker is where an object in placed on an image and it deals with coordinates input(x, y) that performs the following task:

- Mark with a specific ID to the specific object of that image.
- When it moves, track its movement and predict new location of the object depends on many frame attributes.

There are several examples of object tracking algorithm such as MOSSE, GOTURN, MedianFlow etc.

3.4 Combining counter, object tracking and object detecting

To gather all three in one there are few steps that needed to be done and we divided them into three phases:

Phase 1- Waiting: In this Phase We wait for the vehicles to enter into our frame and cross our ROI. No vehicle detection and tracking algorithm gets involved on this. Also, there are UP and DOWN counter which respectively indicates the number of vehicles passed at the left and right lane doesn't change.

Phase 2-Detecting: Object detection tells us what is in an image but also where the object is as well. we used Single Shot Detectors and MobileNets to load an object which is pre-trained in detecting network for better accuracy. MobileNets is very efficient for resource constrained devices.

Phase 3-Tracking: The process in which an elementary set object detections, a creation where a specific ID is used for each elementary detection and follow their movement in a video frame with maintaining the objects specific ID called Tracking.

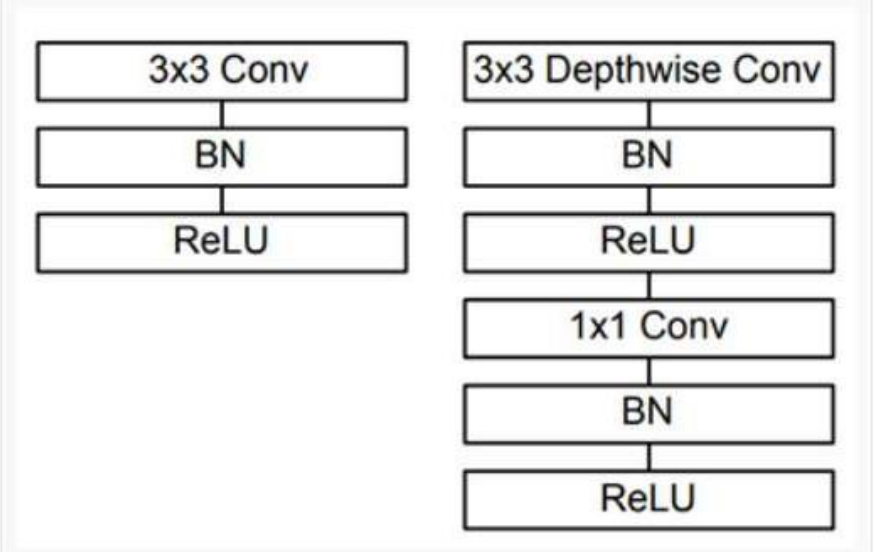


Fig 3.4.1: Mobilenet Architecture.

This hybrid approach has many benefits such as applying exceptionally exact object detection methods. In order to build our vehicle counter, we would need to accomplish a tracker.

3.5 Characteristics of an ideal algorithm

In order to accomplish the programming, we would need to select an algorithm. The algorithm has to:

- Object detection would need one trial only.
- It must work quicker.
- Must be able to operate if the object is lost.
- Able to pick the lost objects from the frame.

3.6 Combining object tracking algorithm

Maintaining the requirements, the best algorithm for this project is Centroid Tracking Algorithm. The reason behind the selection is we would need both OpenCV and "dlib" for tracking vehicles. For standard computer vision, we shall use OpenCV for "dlib" for the implementation of the filter's correlation.

Many research results around the world shows that this algorithm can quickly and effectively detect and track any moving object from video/footage sequence under static background. Here is a brief overview in steps of why this algorithm is suitable for this project:

Step-1: In step-1, it accepts bounding boxe's set and simultaneously calculate their respective centroids (for example: Fig 3.6.1):

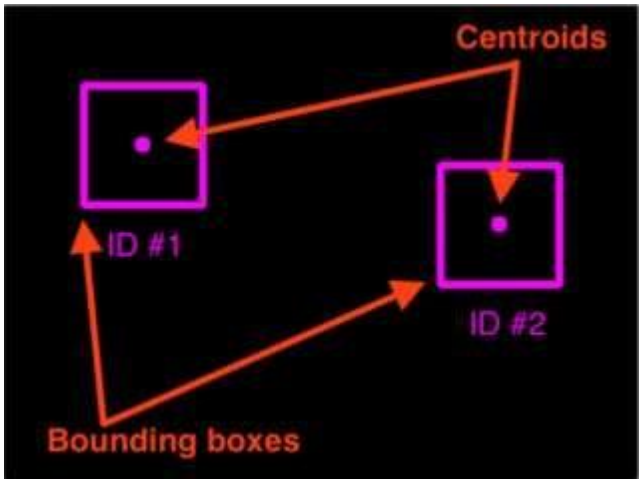


Fig 3.6.1: Bounding boxes.

The bounding boxes can be inflicted in ways like:

- Getting an object detector (e.g SSDs, Faster R CNN etc.)
- Getting an object tracker (e.g MedianFlow filters).

In fig 3.6.1, There are two objects in the beginning of the algorithm.

Step -2: In step-2, There are already centroids (purple and yellow) in the frame. It calculates a distance known as Euclidean Distance between the new and existing centroids (yellow is signed for new and purple is signed for existing centroid).

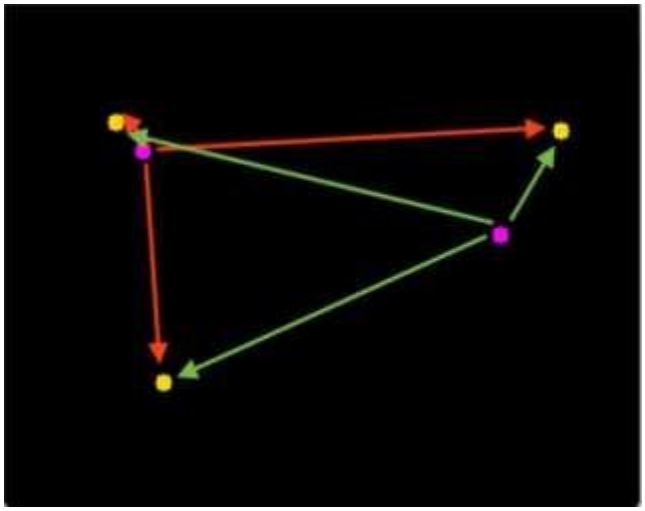


Fig 3.6.2: Euclidean distance.

This algorithm makes the reception that it has to be the same object ID for the dyads with minimum Euclidean distance.

In Fig 3.6.2, There are three new centroids (yellow) and two existing centroids (purple) implying in a way that new one is detected.

Step -3: After Euclidean distance calculation, it attempts to associate with object ID's.

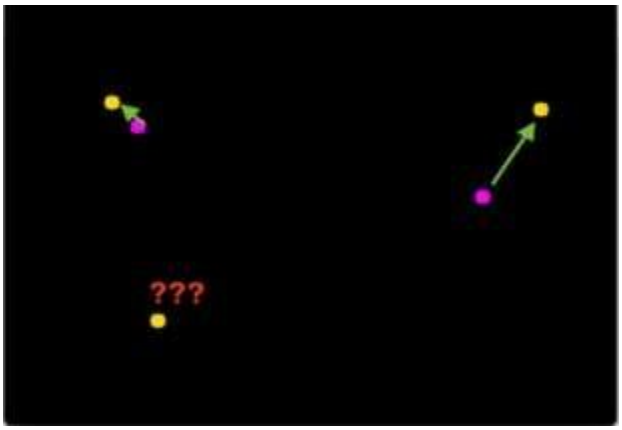


Fig 3.6.3: associating object's ID.

In Fig 3.6.3, we see our tracker has chosen to associate with the centroids in a way that it minimizes their respective Euclidean distances.

Now a question raises that what happens to the point that did not get associated. The answer lies in step-4.

Step -4: In step-4, it registers new objects. Registering new objects means register new object to our track list. We can simply do that by:

- Engaging a new ID or
- Calculating the coordinates of the new object and store the bounding box's centroid.

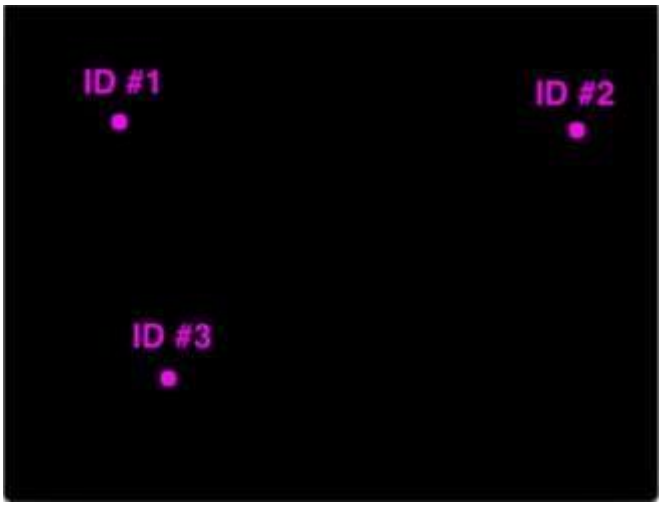


Fig 3.6.4: Registering new objects.

In this process, if an object gets lost or left field, it can be easily deregistered at a new stage. But this is really depend on the exact application of how it is gone missing or invisible.

But anyhow we will deregister the vehicle ID's for our vehicle counter if the don't peers to any existing vehicle for more or equal to 40 consecutive frames.

The way the algorithm tracks object is the reason why we have chosen this for our project.

3.7 Algorithm analysis

We will be using Confusion Matrix for result analysis. The reason behind is, this matrix relatively easy to understand and with its help the classifier's performance can be easily understandable. The only disadvantage is that the terminology can be confusing in many places.

To explain this there's an example of confusion matrix (although it can be done with more than two classes like our project):

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN=50	FP=10	60
Actual: YES	FN=5	TP=100	105
	55	110	

Fig 3.7.1: an example of Confusion Matrix table.

The binary classifier describes:

- **Accuracy:** How accurate the classifier is?

  (TP+TN)/total = (100+50)/165 = 0.91

- **Error Rate:** How inaccurate the classifier is?

  (FP+FN)/total = (10+5)/165 = 0.09

- **True Positive Rate:** How frequently it predicts yes if it's truly yes?

  TP/actual yes = 100/105 = 0.95

- **False Positive Rate:** How frequently it predicts yes if it's truly no?

  FP/actual no = 10/60 = 0.17

- **True Negative Rate:** How frequently it predicts no if it's truly no?

  TN/actual no = 50/60 = 0.83

- **Precision:** Is it corrects if it predicts yes?

  TP/predicted yes = 100/110 = 0.91

- **Prevalence:** How frequent yes occurs?

  actual yes/total = 105/165 = 0.64

3.8 Challenges

We faced a lot of challenges in this section of our research. We only face problem in the research part but also in the coding part as well. We faced a lot of problems like-

- It took a long time to find out which work related to our field has been done before our project topic.
- As we have chosen a higher programming language, there were not enough programming material references that we can take help from.
- There were many errors while writing the scripture.
- We faced difficulties while implementing the programming along with footages.
- It required fully corrected coding otherwise it wouldn't run, therefore we couldn't view any progress

CHAPTER 4
IMPLEMENTATION AND EXPERIMENTAL RESULT

4.1 Introduction

In chapter four, we will discuss the descriptive analysis of our project. Here in this chapter we will state our implementation and experimental result. The chapter closes with the summarization of result.

4.2 Creating trackable object the code

We need to track and count objects from a footage. For that we need :

- Object ID and
- The previous centroids (for computing the direction the object's movement).
- Whether the object has already been counted or not.

To complete the tasks, we need to write the code including "TrackableObject":

```
class TrackableObject:
    def __init__(self, objectID, centroid):
        # store the object ID, then initialize a list of centroids
        # using the current centroid
        self.objectID = objectID
        self.centroids = [centroid]

        # initialize a boolean used to indicate if the object has
        # already been counted or not
        self.counted = False
```

Fig 4.2.1: Code for trackable object

The TrackableObject is a constructor. It only concedes object ID and centroid. When it gets them, it simply stores them combined. It also holds an object's centroid location history. Which is why it is a list.

4.3 Implementing counter with OpenCV and Python

By the help of all library and tools, we are now to ready to implement main coding part.

Fig 4.3.1: Importing libraries.

We started by importing our necessary packages like:

- we imported our custom “CentroidTracker” and “TrackableObject” classes.
- For computing FPS and work with a webcam, the “VideoStream” and “FPS” modules from “imutils.video” .
- “imutils” is required for the OpenCV functions in order to count vehicle number.
- “dlib” is required for tracking. It’s also a library function.

Then we would need to parse command line arguments. After the script’s dynamically handle command line arguments accomplishment at runtime, we would need to prepare our SSD.

```
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
    help="path to optional input video file")
ap.add_argument("-o", "--output", type=str,
    help="path to optional output video file")
ap.add_argument("-c", "--confidence", type=float, default=0.4,
    help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip-frames", type=int, default=30,
    help="# of skip frames between detections")
args = vars(ap.parse_args())


CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]


print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

Fig 4.3.2: Parsing arguments.

Here, there are these arguments (confidence argument, prototxt argument, skip-frames argument, output argument, model argument and input argument) are lined up. We use these arguments for giving facts to the vehicle counting part of the code.

Now for the SSD, we would initiate Classes

We need to fix the CLASS list first. Then we shall go for executing the lists that our SSD selects. It is the model provided in the “Downloads” which is why This list should not be changed.

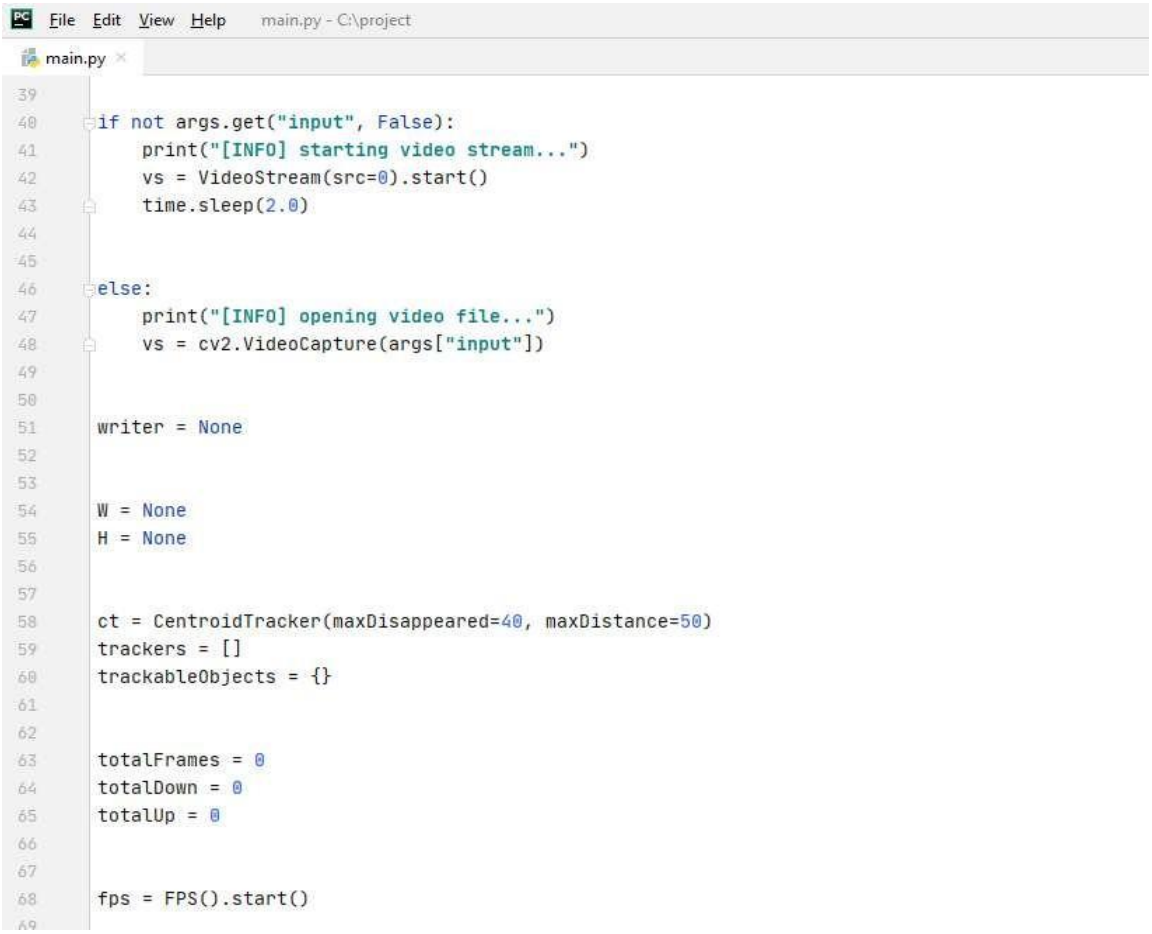


Fig 4.3.3: Initializing Classes.

Where we are using a video/webcam stream, we need to handle the case there (from line 40-43). If not, it shall a video footage to capture the frame that has described on Line 46 to 48.

Before that we have lot to perform. There’s remain “writer” which is our video writer function, “W” and “H” are frame dimensions, “ct” is the centroid tracker, “trackableObjects” maps objectID, “totalFrames” calculates frame numbers., “totalDown” and “totalUp” describes total vehicles that headed downwards or upwards in a FPS.

If the program gets lost in the while loop, it’s time to loop over incoming frames:

```
70
71  while True:
72
73      frame = vs.read()
74      frame = frame[1] if args.get("input", False) else frame
75
76
77      if args["input"] is not None and frame is None:
78          break
79
80
85      if W is None or H is None:
86          (H, W) = frame.shape[:2]
87
88
89      if args["output"] is not None and writer is None:
90          fourcc = cv2.VideoWriter_fourcc(*"MJPG")
91          writer = cv2.VideoWriter(args["output"], fourcc, 30,
92              (W, H), True)
93
94
```

Fig 4.3.4: Code for loop.

The loop starts at Line 71. Before starting the loop we have selected the next frame in Line 73 and 74. Once we have reached the video in that event, it would break out of the loop in line 77 and 78. After that on Line 81 and 82, Pre-processing frame takes place which includes a RGB image. It would need to take a dimension of the frame for the video writer which locates in Line 85 and 86. Now, it would initiate the video writer if there's an output path given (Lines 89 to 92).

4.4 Vehicle Detection

After the looping part, now it's time to detect vehicle using SSD.

```
94
95      status = "Waiting"
96      rects = []
97
98
99      if totalFrames % args["skip_frames"] == 0:
100
101         status = "Detecting"
102         trackers = []
103
104
105         blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
106         net.setInput(blob)
107         detections = net.forward()
108
109
110         for i in np.arange(0, detections.shape[2]):
111
112             confidence = detections[0, 0, i, 2]
113
114
115             if confidence > args["confidence"]:
116
117                 idx = int(detections[0, 0, i, 1])
118
119
120                 if CLASSES[idx] != "car" and CLASSES[idx] != "bus":
121                     continue
```

Fig 4.4.1: Inserting SSD.

We would initiate the status as Waiting in Line 95. It states :

Waiting: Here we are waiting for a vehicle to arrive.

Detecting: As soon as a vehicle arrives, it detects it.

Tracking: Now that Vehicle is found, it is being tracked. The Counnter "totalUP" and "totalDown" are counting vehicles that are going in the up and down directions.

Our "rects" list will be counted as detection or tracking. Then We would go ahead and then initiate "rects" in Line 96.

4.5 Compute a bounding box

Computing bounding box occurs in Line 124 and 125. Then we initiate the tracker resulting as line 129 and 130 resulting in Line 128 to 133.

In the else block, instead of detecting it shall track the object on Line 138 to 173.

From there we took the position coordinates revealed by calculating the information in our "rects" list.

```
                box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
                (startX, startY, endX, endY) = box.astype("int")

            box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
            (startX, startY, endX, endY) = box.astype("int")


            tracker = dlib.correlation_tracker()
            rect = dlib.rectangle(int(startX), int(startY), int(endX), int(endY))
            tracker.start_track(rgb, rect)


            trackers.append(tracker)


else:

    for tracker in trackers:

        status = "Tracking"


        tracker.update(rgb)
        pos = tracker.get_position()


        startX = int(pos.left())
        startY = int(pos.top())
        endX = int(pos.right())
        endY = int(pos.bottom())


        rects.append((startX, startY, endX, endY))
```

Fig 4.5.1: Computing bounding boxes.

```
154
155
156     cv2.line(frame, (0, H // 2), (W, H // 2), (0, 255, 255), 2)
157
158
159     objects = ct.update(rects)
160
```

Fig 4.5.2: Code for horizontal line drawing.

Now, we would a horizontal visualization line on Line 156 and 159 so that if a vehicle passes it gets detected on the line.

```
160
161
162     for (objectID, centroid) in objects.items():
163
164         to = trackableObjects.get(objectID, None)
165
166
167         if to is None:
168             to = TrackableObject(objectID, centroid)
169
167         if to is None:
168             to = TrackableObject(objectID, centroid)
169
170
171         else:
172
173             y = [c[1] for c in to.centroids]
174             direction = centroid[1] - np.mean(y)
175             to.centroids.append(centroid)
176
177
178             if not to.counted:
179
180                 if direction < 0 and centroid[1] < H // 2:
181                     totalUp += 1
182                     to.counted = True
183
184
185                 elif direction > 0 and centroid[1] > H // 2:
186                     totalDown += 1
187                     to.counted = True
188
189
190         trackableObjects[objectID] = to
```

Fig 4.5.3: Code for tracking vehicle movement.

In this next block (Line 162-90) we would figure whether or not a vehicle moved up or go down in the frame. On line 164, we have fetched “TrackableObject” for current “ObjectID”. If it fails, we create a new one on line 167 and 168.

Else there is an existing “TrackableObject” that needed to be figured out if the object(vehicle) is moving up or down.

Then, we would store the values of the attributes in dictionary on Line 190. Now it will capture and if needed will update it afterwards.

4.6 Frame for visualization

```
        text = "ID {}".format(objectID)
        cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
        cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)


    info = [
        ("Up", totalUp),
        ("Down", totalDown),
        ("Status", status),
    ]


    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)


    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)


    if writer is not None:
        writer.write(frame)


    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF


    if key == ord("q"):
        break


    totalFrames += 1
    fps.update()


fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))


if writer is not None:
    writer.release()


if not args.get("input", False):
    vs.stop()


else:
    vs.release()


cv2.destroyAllWindows()
```

Fig 4.6.1: Code for frame for visualization.

We would need to draw some information for visualization:

- ObjectID - every object's needed to have an identifier and it has be numeric.
- Centroid - Object's centre will be represented by a "dot".
- info - it covers "totalUp" , "totalDown" , and "status"

In the next block we write the "frame" on Line 212 and 213, Display the "frame" and also handles keypresses from Line 216 to 221 which breaks if "q" is pressed, update "fps" on Line 225.

Now at the end, we shall close any open windows and release all pointers and open the command prompt.

After 245 Line of codes, We are done here writing the scripture. Now it's showing the output time.

4.7 Experimental result

To see the result, we would need to open the command prompt.

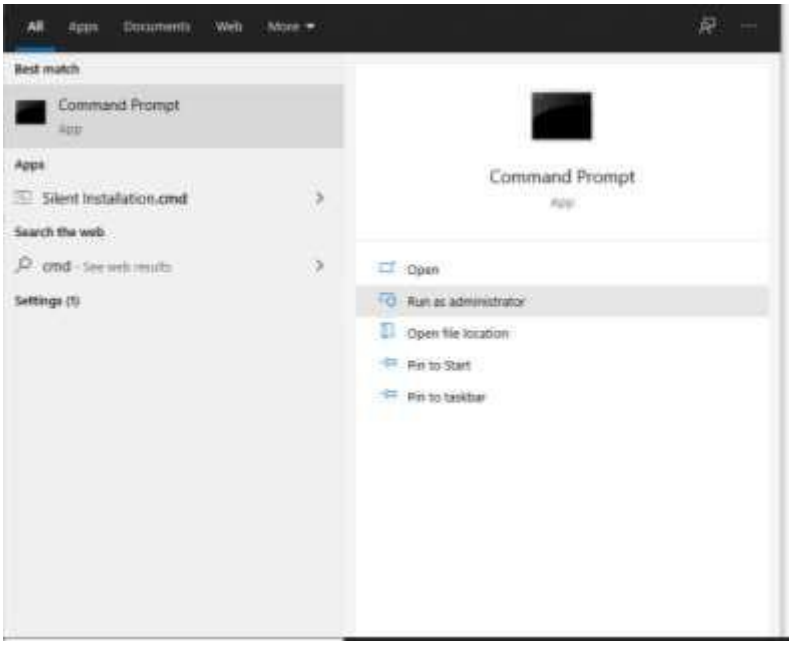


Fig 4.7.1: Selecting Command Prompt.

After opening command prompt, we need to type our project name

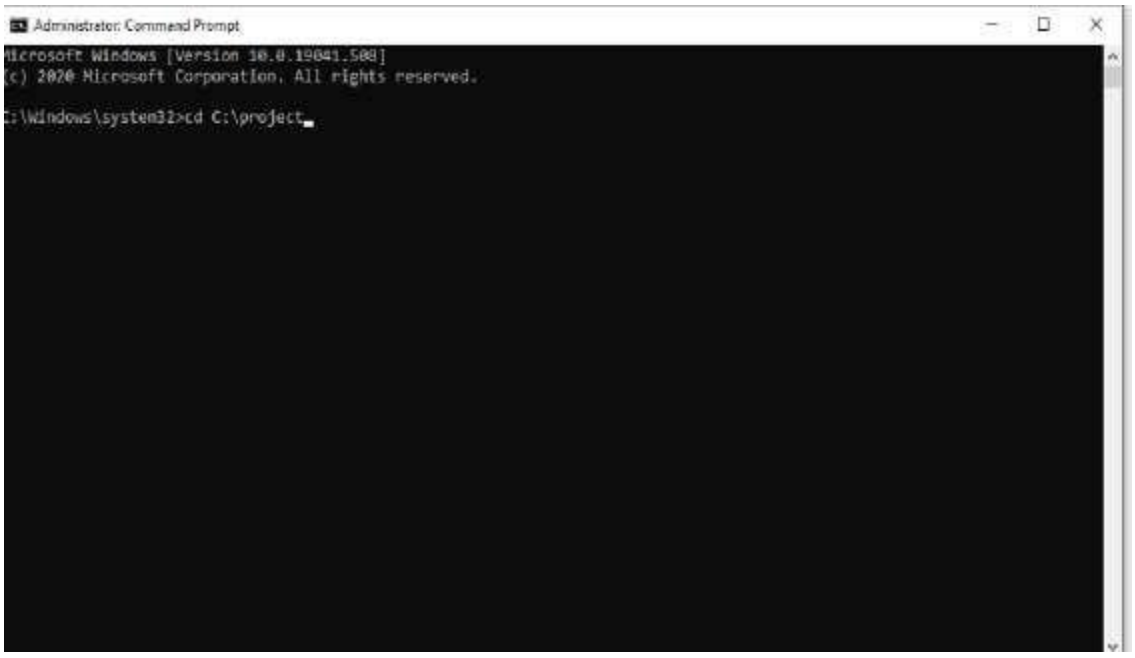


Fig 4.7.2: Selecting project folder.

After that a window will appear showing vehicle detecting, tracking and counting.

Fig 4.7.3: Output 1.

Fig 4.7.4: Output 2.

4.8 Discussion

We have implemented necessary python libraries, object tracking algorithm, created trackable object, visualized the frame and it was a successful trial. We were able to assemble all those to work together collectively. We also have faced lots of difficulties to produce the given figures which represents our project's experimental results. The result was quite expected after all of the hard work. Our project works effectively.

CHAPTER 5
CONCLUSION AND THE FUTURE SCOPE

5.1 Conclusion

An organized traffic management is a blessing to every citizens of a nation. Providing Traffic Flow information could be the medium for that blessing through an easy way. Our project purpose was a small try to the contribution. Based on state-of-the-art of video-based supervision systems, we have successfully built a program which can execute our project principle. Our project can detect, track and count every vehicle movement. The system we have developed is affirmed as efficient and robust.

5.2 Future Scope

This research can be implemented and further developed with the collaboration of any governmental support. Because with the severe growth of population, traffic jam has become one of the major problems in our modern life. So, the citizens will be more interested where they get facilities about traffic flow information. Our system can provide great facilities for the common people as they will be more interested in utilizing their time. We have tested this multiple times. Currently our project is limited to a program file. But in the near future, we were planning to convert it in an application with some specific features in a higher level such as we can create our own application system where people in Bangladesh can use it like Uber/Pathao/Shohoz app.

REFERENCE

[1] Bommes, M.; Abdulrahim, K., and Oeser, M. (2016) "Video-based Intelligent Transportation Systems – State of the Art and Future Development," Proc. Of the 6th Transport Research Arena, Warsaw, pp. 225-230.

[2] Hoffman, C. and U. Regensburger (2004) "Vehicle Detection Fusing 2D Visual Features", IEEE International Vehicles Symposium, pp. 220.

[3] Learn about Digital Image, available at << https://en.wikipedia.org/wiki/Digital_imaging />>, last accessed on 19-09-2020 at 7:15 AM.

[4] D. M. Jang and M. Turk, " Vehicle Recognition Using Curvelet Transform and SVM," in Applications of Computer Vision pp 415-425.

[5] Yoshua Bengio, H. King, Class Lecture, Topic: "Image processing and visualization.", College of Engineering and Science, Victoria University, Melbourne, Apr. 2016..

[6] Learn about Vehicle Detection with Bluetooth based automatic detection technique and, available at << https://www.hindawi.com/journals/wcmc/2019/3159762/>>last accessed on 28-04-2020 at 10:00 PM.

[7] Chen S. R. Palacios Zhu and Xiong, G. "Parallel Transportation Management and Control System and Its Applications in Building Smart Cities," IEEE Transactions on Intelligent Transportation Systems, pp. 180-183.

[8] Learn about How Augmented Reality Is Redefining the Fashion Industry, available at https://www.ukessays.com/essays/computer-science/image-processing-for-future-applications-computer-science-essay//>>, last accessed on 11-02-2019 at 11:45 AM.

[9] Learn about Track Algorithms, available at << https://en.wikipedia.org/wiki/Track_algorithm/>>, last accessed on 10-08-2020 at 10