REVA UNIVERSITY
Bengaluru, India

A Project Report on

Smart Parking Solution for Multi Building Campus

Submitted in Partial Fulfilment for Award of Degree of
Master of Technology
In Artificial Intelligence

Submitted By
Rajanna
R20MTA07

Under the Guidance of
Ravi Shukla
Mentor, Artificial Intelligence, RACE,REVA University

REVA Academy for Corporate Excellence - RACE
REVA University
Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru - 560 064
race.reva.edu.in

August, 2022

REVA
UNIVERSITY
Bengaluru, India

Candidate's Declaration

I, **Rajanna** hereby declare that I have completed the project work towards the Master of Technology in Artificial Intelligence at, REVA University on the topic entitled **"Smart Parking Solution for Multi Building Campus**" under the supervision of **Mr. Ravi Shukla, Mentor, Artificial Intelligence, RACE, REVA University.** This report embodies the original work done by me in partial fulfilment of the requirements for the award of degree for the academic 2022**.**

Place: Bengaluru	Name of the Student: Rajanna
Date: 27.08.22	Signature of Student

REVA UNIVERSITY
Bengaluru, India

Certificate

This is to Certify that the project work entitled **Smart Parking Solution for Multi Building Campus** carried out by **Rajanna** with **SRN R20MTA07** is a bonafide student of REVA University, is submitting the project report in fulfilment for the award of Master of Technology in Artificial Intelligence during the academic year 2022. The Project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 15%. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said degree.

Ravi Shukla.

Signature of the Guide
Ravi Shukla

Signature of the Director
Dr. Shinu Abhi

External Viva Panelists

Names of the Examiners

1. Dr. Sai Hareesh, Research Expert, SAP Labs India
2. Pradeepta Mishra, Director – AI, L&T InfoTech

Place: Bengaluru

Date: 27.08.22

REVA UNIVERSITY
Bengaluru, India

Acknowledgement

I am highly indebted to **Dr. Shinu Abhi**, Director, Corporate Training for the guidance and support provides throughout the course and my project.

I would like to thank **Mr. Ravi Shukla** for the valuable guidance provided as my project guide to understand the concept and in executing this project.

It is my gratitude towards our Mentor, **Dr. Jay Bharateesh Simha**, and all other mentors for the valuable guidance and suggestions in learning various data science aspects and for the support. I am grateful to them for their valuable guidance on several topics related to the project.

I am thankful to my classmates for their support, suggestions, and friendly advice during the project work.

I would like to acknowledge the support provided by the founder and Hon'ble Chancellor, **Dr. P Shayma Raju**, Vice-Chancellor, **Dr. M. Dhanamjaya**, and Registrar, **Dr. N Ramesh**.

It is sincere thanks to all members of the program office of RACE who were always supportive in all requirements from the program office. It is my sincere gratitude towards my parents and my family for their kind co-operation. Their encouragement also helped me in the completion of this project

Place: Bengaluru
Date: 27.08.22

REVA UNIVERSITY
Bengaluru, India

Similarity Index Report

This is to certify that this project report titled **Smart Parking Solution for Multi Building Campus** was scanned for similarity detection. Process and outcome is given below.

Software Used: Turnitin
Date of Report Generation: 26.08.2022
Similarity Index in %: 11%
Total word count: 4202
Name of the Guide: Ravi Shukla

Place: Bengaluru
Date: 27.08.2022
Verified by: M N Dincy Dechamma

Name of the Student: Rajanna
Signature of Student

Signature
Dr. Shinu Abhi,
Director, Corporate Training

List of Abbreviations

Sl. No	Abbreviation	Long Form
1	RFID	Radio Frequency Identification
2	GMM	Gaussian Mixture Model
3	BS	Background subtraction
4	CNN	Convolutional Neural Networks
5	DL	Deep Learning
6	ML	Machine Learning
7	CV	Computer Vision
8	YOLO	You Only Look Once

List of Figures

List of Tables

Abstract

Locating parking space is becoming a serious problem as more and more employees prefer to switch to personal commute due to prevailing pandemic situation across globe in office campus. Particularly, where there is multiple building inside main campus, and it is quite challenging to keep track of car parking inside a given building. Without parking availability information, commuter must shuttle around builds to find parking space during peak hours, and this will be a loss of time as well as frustration to the commuter.

There are few options companies follow like physically counting cars at certain interval and update the dashboard, install badge reader at entry and exit and people have to punch card at both the places, sensors have been installed in certain parking lots, allowing management to track when cars enter and exit, install Radio Frequency Identification (RFID) and reader at entry and exit in order to track the capacity. All these methods will bring in burden on additional resource and high cost of installation of reader & necessary barricade systems respectively. Also, will add to regular maintenance of overall system.

The proposed solution is implemented using Deep Learning (DL) which is popular machine learning algorithm widely used in many areas including traffic monitoring that cameras placed at strategic locations on roadways can perform a wide variety of functions, including counting and identifying vehicles, monitoring for traffic violations and speeds, and more. The proposed solution is achieved by counting vehicle at entry and exit, thus the available parking capacity as a dashboard at main entry of the building. This project details DL implementation to create a vehicle counting system by using video from camera installed at the entry and exit path for vehicle. Pretrained model of YOLOv4 is used for object detection, vehicle tracking with Euclidean distance and counting. Since vehicle movement is considerably slow in pathway, performance of the model tested is 96.8% to 100% for varied frame rate of the video in parking capacity tracking.

Keywords: Deep learning, Yolov4, Object detection, Vehicle counting, RFID, Parking

Table of Contents

Chapter 1: Introduction

The rising expense and increasing difficulty of parking is becoming a big issue for cities. It's difficult, for instance, to get a parking spot near the office. Users are more likely to be annoyed by the process of looking for a parking spot in a multibuilding campus parking lot. For example, it can be difficult to locate a parking spot on busy event days like game day or annual day. People driving around looking for parking spots waste time and money. Finding a parking spot in a parking garage or lot can be a major hassle for drivers. This annoyance might be caused either by the scarcity of parking spots or by the fact that another car has already occupied them. As a result, we'll need to construct parking solutions that are both smart and novel if we're going to be able to meet the demand. Some parking garages have guiding systems that use sensors to alert cars when a spot becomes available. One such setup uses sensors placed in each parking space in conjunction with a light visible from a distance. Once a car is parked in the space, the light will automatically switch off. Thus, motorists can save time by quickly locating open parking spots without having to drive around the entire parking lot. This simple technique is helpful for vehicles who have already arrived in the parking lot, but it is of no use to those who are still on their way.

In recent years, numerous proposals have been made for improved parking guidance systems. There are benefits and drawbacks to each of these modern systems. A method of determining whether or not a car is parked in a given space is essential to all of these systems. This mechanism could be anything from a straightforward ultrasonic sensor [1] that detects a vehicle at a threshold distance to a sophisticated optical sensor [2] that activates dependent on distance. The presence of a vehicle or other object is detected by these sensors. The system must have a method of alerting drivers or indicating that a parking space is taken after a car has been seen. The automobiles themselves are expected to be able to transmit and receive data in some systems [3][4]. The term Vehicular Ad Hoc Network (VANET) describes a system in which cars can exchange data with one another and with sensors in the surrounding environment.

Many research have been conducted for traffic management applications based on image and video processing approaches. The analysis of traffic video data includes detection or recognition of vehicles, measurement of vehicle's speed, generation of tracking trajectory, counting of vehicles, congestion of traffic and collisions of vehicles. These applications have become popular recently due to the availability of low-cost cameras and embedded devices. Thus, the video data analytic is one of the prime research focuses in the computer vision and big-data area.

When compared to traditional Machine Learning (ML) methods, Deep Learning (DL) performs better in many scenarios, particularly those involving Computer Vision (CV). To learn the pattern of objects or photos and identify the object that is taken by camera, ML plays an important part in CV. Detecting, classifying, or recognizing objects inside an image using ML algorithm used to necessitate a preprocessing and feature extraction step in a CV system [5] [6]. Preprocessing and feature extraction methods need to be tailored to the specific item or circumstance at hand. Because of this, conventional CV's one-model approach can only identify or recognize a limited set of objects. In contrast, DL can automatically preprocess and extract the picture features inside its networks, classify the image class, and locate the location of each object within the image thanks to its massive and deep networks. However, DL calls for machines with very particular hardware requirements, as well as a massive amount of data to properly train the networks and maximize their effectiveness.

This work proposes an easy-to-use system for counting vehicles based on the deep learning algorithm. A well-known pretrained YOLOv4 DL architecture known for its excellent accuracy in object detection and its moderate computation compared to other DL architectures. Vehicle tracking with Euclidean distance and counting entry and exit vehicles for accurate for parking capacity tracking.

Chapter 2: Literature Review

In recent years, numerous proposals have been made for improved parking guidance systems. There are benefits and drawbacks to each of these modern systems. A method of determining whether or not a car is parked in a given space is essential to all of these systems. This mechanism could be anything from a straightforward ultrasonic sensor [1] that detects a vehicle at a threshold distance to a sophisticated optical sensor [2] that activates dependent on distance. The presence of a vehicle or other object is detected by these sensors. The system must have a method of alerting drivers or indicating that a parking space is taken after a car has been seen. The automobiles themselves are expected to be able to transmit and receive data in some systems [4][5]. If the vehicle itself can communicate with sensors and with other vehicles.

Another cloud-based [7] system that uses a mobile application that is linked to the cloud has been implemented. The user will decide what time he wants to assign the space. The user will be issued the alarm if he didn't occupy it afterwards. The number of reserved and vacant parking spaces will be displayed on the app.

The counting of vehicles is often performed by splitting the scene into a moving foreground and a stationary background. Background subtraction (BS) and blob analysis [8] [9] [10], BS combined with Gaussian Mixture Model (GMM) based BS [9] [11] , and particle filter based tracker [12] [11] . are used to achieve this goal. Frame averaging, the Gaussian mixture model, and principal component images are common methods for achieving BS [13] However, when dealing with traffic data, the background removal method has its limits. Partial occlusion in processed picture data causes vehicles to merge, and inaccurate bounding box predictions are made. Additionally, car shadows lead to erroneous detection. The results are enhanced by combining a Gaussian mixture model and an expectation maximization to create the background model. The moving vehicles are then retrieved from the background using subtraction. Morphological features and color histograms are used to deal with the occlusions. Extracting the bright patches (i.e., headlights), then pairing them, monitoring them, and finally counting them, is a proposed method for counting vehicles in nighttime [14]. Feature extraction and a classifier-based method are used in the methods provided in [15] for counting vehicles. One application of Computer Vison (CV) technology is in traffic monitoring systems, which can be used to keep tabs on things like vehicle speeds, parking lot occupancy rates, and traffic violations. The first step in every one of the aforementioned chores is finding where each car now resides. Therefore, an object detection method is so important. Preprocessing techniques like grayscale picture scaling, binarization of images, and background reduction[16] [17] [8] or even edge detection are required by the traditional Machine Learning approach to accomplish this task. There are, of course, drawbacks to this method; for example, if the vehicle's shadow is present in the image, the detection may be less accurate. Changes to the road surface, such as road maintenance, road damage, or any barriers on the road, can also cause inaccurate detection since they can disrupt the image subtraction process.

Although it is computationally expensive and requires a considerable amount of data to train the networks, the DL methodology provides more adaptable performance without the requirement to preprocess the image and extract the feature using multiple methods. There are even more advanced DL architectures available, which have been trained with millions of data points, making the creation of CV systems much simpler.

It is well-known that the Convolutional Neural Networks (CNN) architecture of DL performs exceptionally well in Computer Vision (CV), particularly in the tasks of object detection and categorization. Regional-based CNN (R-CNN), Fast R-CNN, Faster R-CNN, Region-based Fully CNN (R-FCN), Single Shot Detector (SSD), Mask R-CNN, YOLO, YOLOv2, YOLOv3, and YOLOv4 are only some of the CNN-based architectures used in CV for object recognition and categorization. For this experiment, YOLOv4 was chosen as the best model to utilise because of its high accuracy and fast computation time in real time compared to the other object identification methods.

Chapter 3: Problem Statement

Locating parking space is becoming a serious problem as more and more employees prefer to switch to personal commute due to prevailing pandemic situation across globe in office campus. Particularly, where there is multiple building main campus, and it is quite challenging to keep track of car parking inside a given building. Without parking availability information, employees must shuttle around builds to find parking space during peak hours, and this will be a loss of time as well as frustration to the commuter.

There are few options companies follow like physically counting cars at certain interval and update the dashboard, install badge reader at entry and exit and people have to punch card at both the places, sensors have been installed in certain parking lots, allowing management to track when cars enter and exit., install Radio Frequency Identification (RFID) tag on each employee vehicle and reader at entry and exit in order to track the capacity. Very few or negligible companies have installed simple ultrasonic sensor [1] at each parking space and these are hardwired to the centralized system to detect parking occupancy.

All these methods will bring in burden on additional resource and high cost of installation of reader & necessary barricade systems respectively. Also, will add to regular maintenance of overall system.

Chapter 4: Objectives of the Study

This project proposes to use advancement in deep learning robust method You Only Look Once (YOLO) which treats the object detection as a regression problem to map pixels into bounding boxes with class probabilities. Moreover, it computes everything in a single evaluation. Euclidean distance methodology for object tracking and boundary line to count entry or exit of the vehicle in a single frame of the video to capture the parking availability in each building.

The primary objective of this study is to implement the customized smart parking solution for multibuilding campus offices. Having multibuilding it is extremely important to let the employees know where the parking space is available especially during peaking hour. A dashboard at each building entrance encompassing overall occupancy and available status will greatly help the employee peacefully approach the building where parking is available.

Secondary objective of the study helps in selection of optimal video mode for better accuracy for model for vehicle detect and counting.

Chapter 5: Project Methodology

The vehicle counting system which was built in this work has three main modules, i.e. Object Detection Module, Vehicle Tracking Module and Counting Module as given in Figure No. 5.1. The first module reads every single frame from the video and doing vehicle detection using YOLOv4 algorithm. This module results the location of every detected vehicle, i.e. the bounding box coordinates. The second module tracks it's by its centroid location between frames to frame for tracking and the third module counts the number of vehicles that crossed the road depending on the coordinates or position of the vehicles. So, the result of object detection module plays vital function in this system, since if the vehicle is not spotted, then it will not be counted.

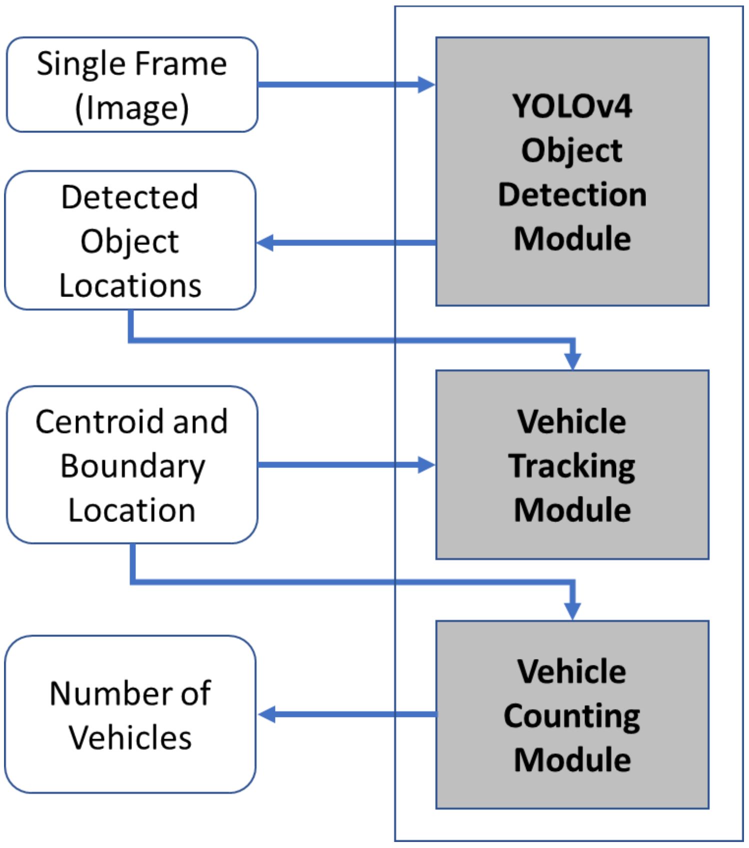

Single Frame (Image)
Detected Object Locations
Centroid and Boundary Location
Number of Vehicles
YOLOv4 Object Detection Module
Vehicle Tracking Module
Vehicle Counting Module

Figure No. 5.1: General architecture of the vehicle counting system.

Video containing entry and exit of the vechicle from singe camera is used to demonistrate the proposed solution.

Chapter 6: Resource Requirement Specification

There are primarily three resources that are need for the proposed solution.

1. Dataset
2. Experimental setup
3. Deployment libraries

6.1 Dataset

Data for the proposed solution is implementation using video from surveillance camera referenced at [18] . Note that all the figures presented in this work are outcome of the model at various stage of the process. This is an 81 second video with 1280 x 720 pixel having 30 frame per second frame rate and same video is converted to 15 frame per second frame rate for the study. This video has two-way traffic by which both entry and exit of vehicle data has been extracted to run through the model. In both the direction 48 cars will pass in enter and exit path each. Specifically, video of different frame rate was taken because more the frame rate means higher storage capacity of storage for surveillance. Hence most of the companies record the video at lower frame per second to reduce the storage space and the cost associated with it. Snap of the video is shown in the Figure No. 6.1.

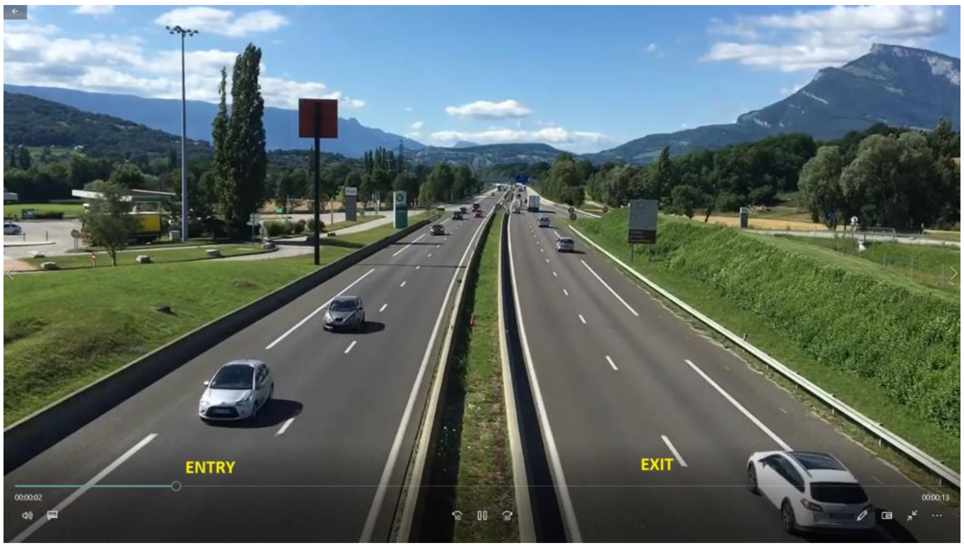


Figure No. 6.1: Single image from a video

6.2 Experimental setup

The proposed solution implemented using OPENCV 4.5.2 and PYTHON 3.8.5 in a machine with Intel(R) Core i5-10210U, CPU of 2.10GHz processor with 16GB RAM. Even though this engine is not as fast as the GPU enabled system, but sufficient to run the deep learning.

6.3 Deployment libraries

The following are the requirements for this resource.

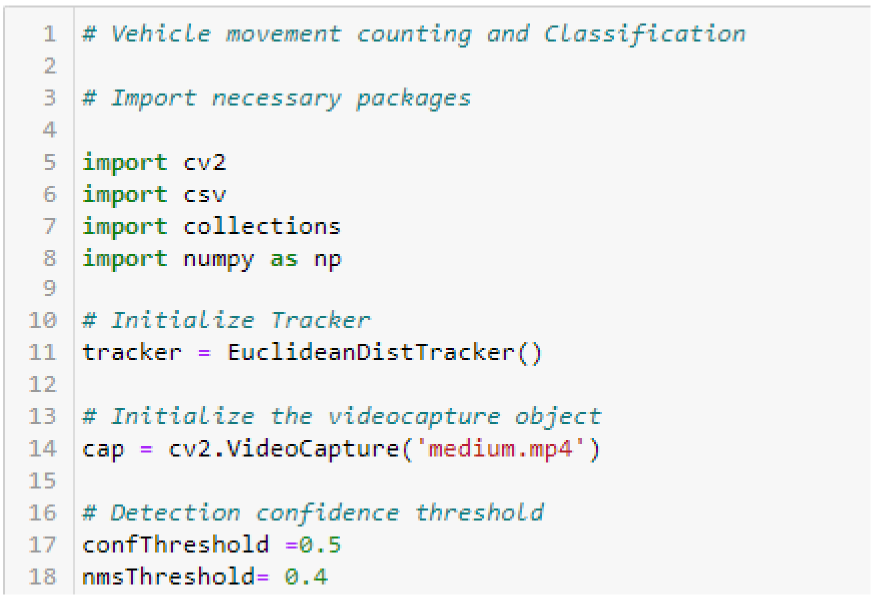
Python environment with required libraries

- OpenCV for loading the video, writing the dashboard, and drawing the centroid.
- YOLOv4 which is the pre-trained primary module for object detection
- Numpy to work and process with array
- Math for mathematical calculation and Euclidean distance measurement

Chapter 7: Implementation

Implementation steps for the proposed solution discussed below. Broadly it is grouped in to six sections. Model will be test through 81 seconds video with 1280 x 720 pixel having 15 and 30 frame per second frame rate. This video has two-way traffic by which both entry and exit of vehicle data has been extracted to run through the model. In both the direction 48 cars will pass in enter and exit path each.

7.1 Import necessary packages and initialize the network



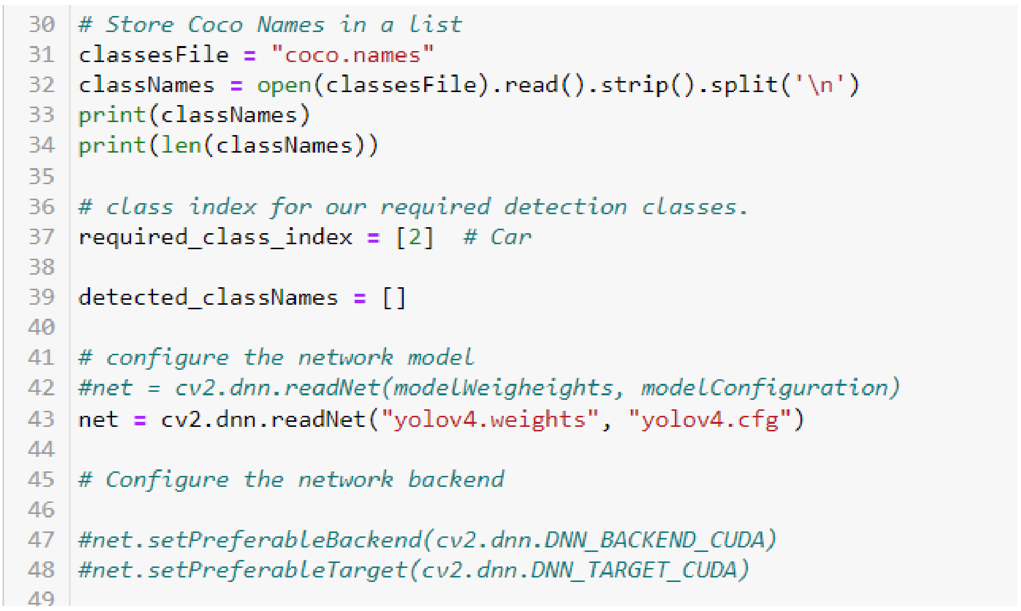
```python
# Vehicle movement counting and Classification

# Import necessary packages

import cv2
import csv
import collections
import numpy as np

# Initialize Tracker
tracker = EuclideanDistTracker()

# Initialize the videocapture object
cap = cv2.VideoCapture('medium.mp4')

# Detection confidence threshold
confThreshold =0.5
nmsThreshold= 0.4
```

Figure No. 7.1: Import necessary libraries

First, import all the necessary packages needed for the project as in the Figure No. 7.1. Then, initialize the *EuclideanDistTracker()* object from the tracker program created next section.

7.1.1 Loading YOLOv4 object detection model

Since YOLOv4 was trained using the COCO dataset, read the file containing all the class names and saved them in a list. There are a total of 80 unique classes in the COCO dataset. Since automobile detection is essential to this project, the *required_class_index* has been set to the index = [2] of the coco dataset's car class. Since this is a CPU-based computer, need to disable the DNN backend as CUDA; if you're working with a GPU, you can comment out those lines as shown in the Figure No. 7.2.

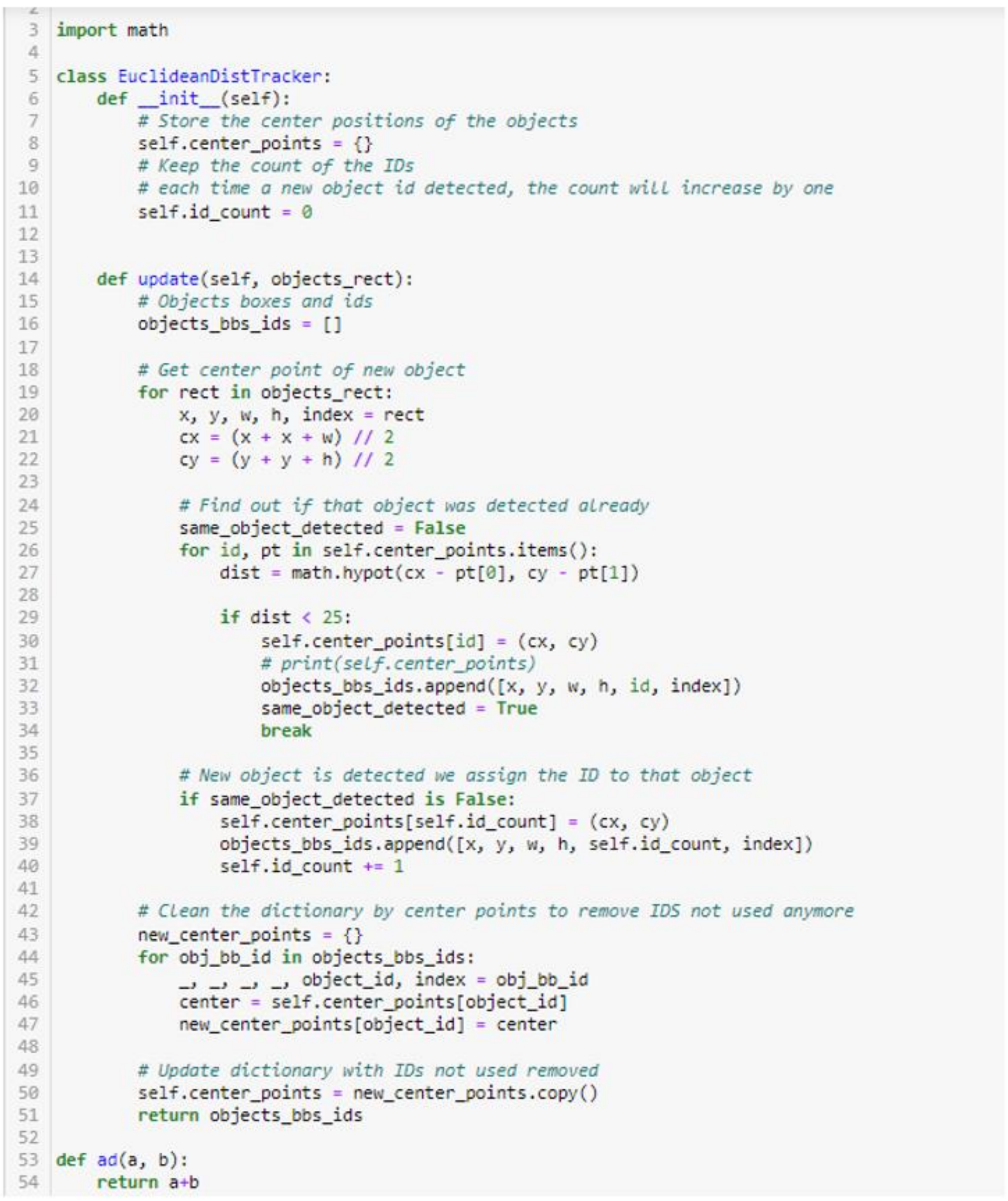


```python
# Store Coco Names in a list
classesFile = "coco.names"
classNames = open(classesFile).read().strip().split('\n')
print(classNames)
print(len(classNames))

# class index for our required detection classes.
required_class_index = [2]  # Car

detected_classNames = []

# configure the network model
#net = cv2.dnn.readNet(modelWeigheights, modelConfiguration)
net = cv2.dnn.readNet("yolov4.weights", "yolov4.cfg")

# Configure the network backend

#net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
#net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

```

Figure No. 7.2: Loading Object detection model

7.1.2 Euclidean distance class function for Object tracking

The core of the tracker's functionality is predicated on the idea of Euclidean distance in Figure No. 7.3. If the distance between an object's centers in the current and prior frames is less than the threshold distance, then the object is confirmed to be the same object from the previous frame. The "*EuclideanDistTracke()r*" class function was developed for tracking, and if the distance between two frames is less than 25, then the two objects being tracked are the same.



```python
import math

class EuclideanDistTracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0


    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h, index = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 25:
                    self.center_points[id] = (cx, cy)
                    # print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id, index])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count, index])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id, index = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids

def ad(a, b):
    return a+b
```

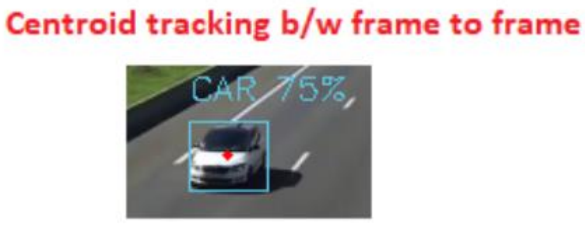
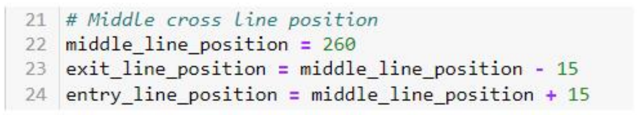


Figure No. 7.3: Creation object tracking module

7.1.3 Vehicle counting module



```
# Middle cross line position
middle_line_position = 260
exit_line_position = middle_line_position - 15
entry_line_position = middle_line_position + 15
```

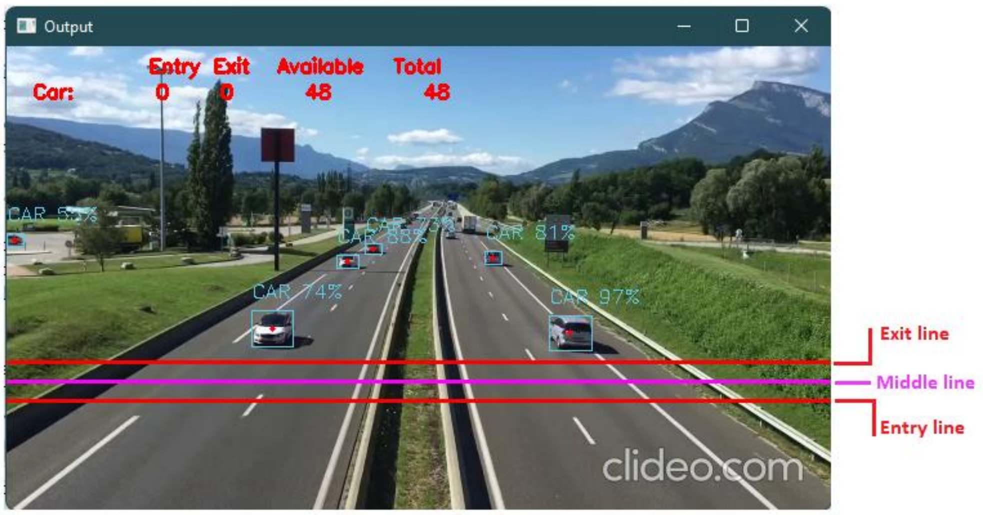
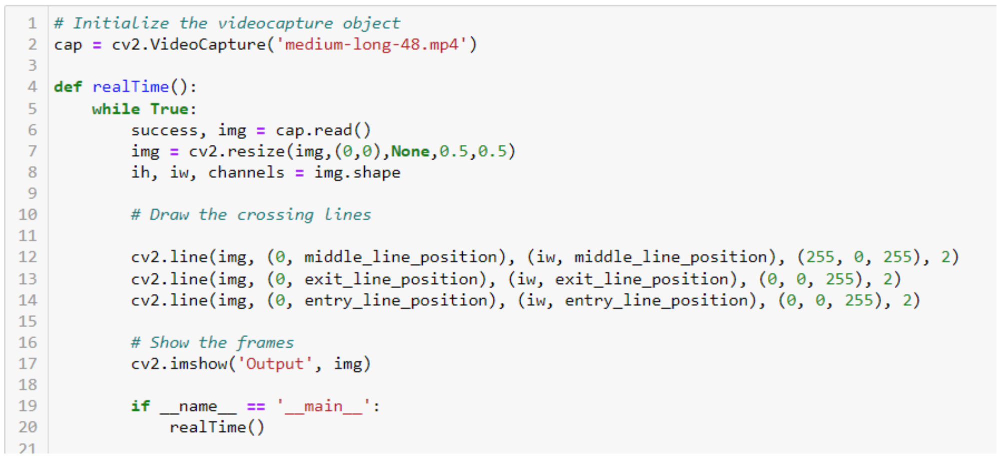


Figure No. 7.4: Creation of vehicle counting module

If the centroid of the vehicle moves from middle line to entry line, considered employee’s car has occupied the parking and if the centroid of the vehicle move from middle line to exit line, considered employee’s car has freed up the parking as represented in the Figure No. 7.4.

7.2 Read frames from a video file

Capture a video using a file reader and a *VideoCapture()* object as mentioned in Figure No. 7.5. When *cv2.reshape()* used, were able to cut the size of our frame in half. Following this, use the *cv2.line()* function to sketch the frame's intersecting lines. The *cv2.imshow()* function was then used to display the output image.

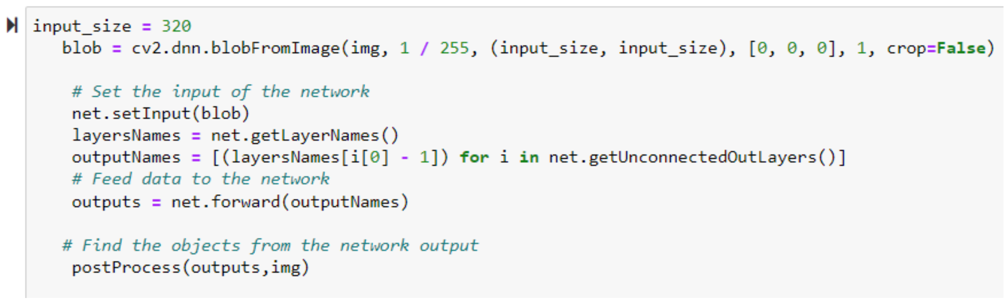


```python
# Initialize the videocapture object
cap = cv2.VideoCapture('medium-long-48.mp4')

def realTime():
    while True:
        success, img = cap.read()
        img = cv2.resize(img,(0,0),None,0.5,0.5)
        ih, iw, channels = img.shape

        # Draw the crossing lines

        cv2.line(img, (0, middle_line_position), (iw, middle_line_position), (255, 0, 255), 2)
        cv2.line(img, (0, exit_line_position), (iw, exit_line_position), (0, 0, 255), 2)
        cv2.line(img, (0, entry_line_position), (iw, entry_line_position), (0, 0, 255), 2)

        # Show the frames
        cv2.imshow('Output', img)

        if __name__ == '__main__':
            realTime()
```

Figure No. 7.5: Reading video file

7.3 Pre-process the frame and run the detection

Our implementation of YOLO can process images with a 320x320 pixel resolution. Blob data is the network's input. If you provide an image to the *dnn.blobFromImage()* function, it will return a shrunk and normalized blob object as called out in Figure No. 7.6. Putting the picture into the network is done with the help of the *net.forward()* function. Plus, there is a result that may be retrieved. Call a user-defined post-processing function, *postProcess()* to post-process the output.

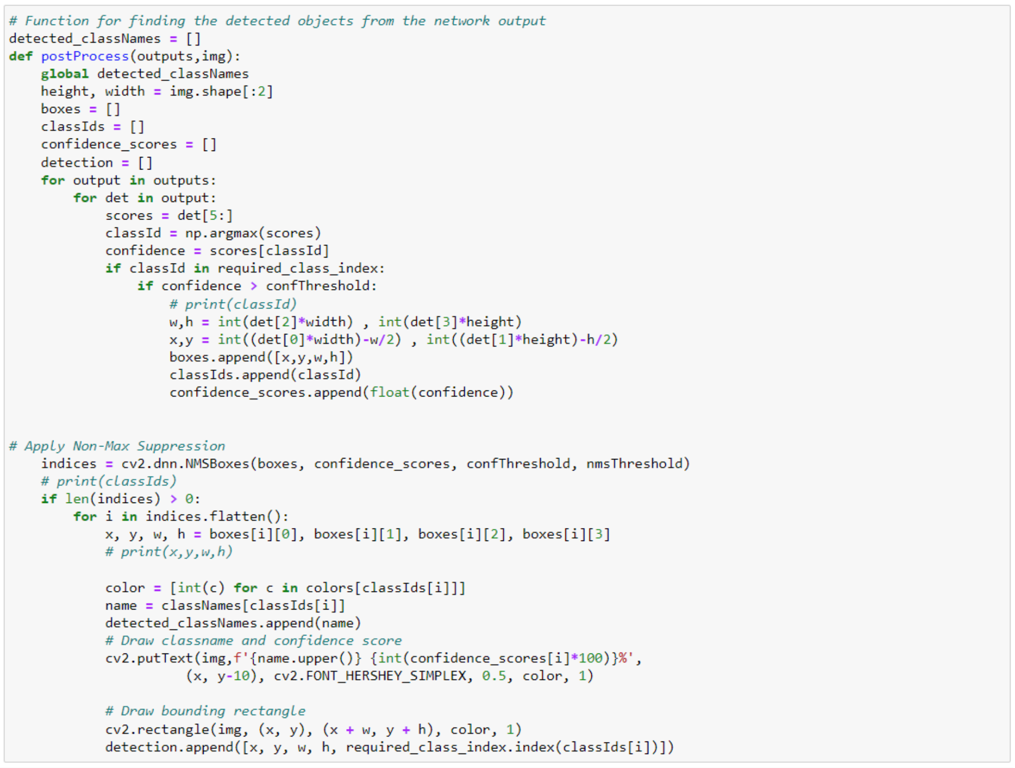


```python
input_size = 320
   blob = cv2.dnn.blobFromImage(img, 1 / 255, (input_size, input_size), [0, 0, 0], 1, crop=False)

    # Set the input of the network
    net.setInput(blob)
    layersNames = net.getLayerNames()
    outputNames = [(layersNames[i[0] - 1]) for i in net.getUnconnectedOutLayers()]
    # Feed data to the network
    outputs = net.forward(outputNames)

   # Find the objects from the network output
    postProcess(outputs,img)
```

Figure No. 7.6: Run object detection

7.4 Post-process the output data

If the class's confidence score is higher than the threshold set in *confThreshold*, the collect its details and save its box co-ordinates, class-Id, and confidence score in three different tables as in Figure No. 7.7. Since YOLO can return numerous bounding boxes for the same item, use the *NMSBoxes()* method to narrow down the results and select the most accurate detection box before drawing any text into the scene.

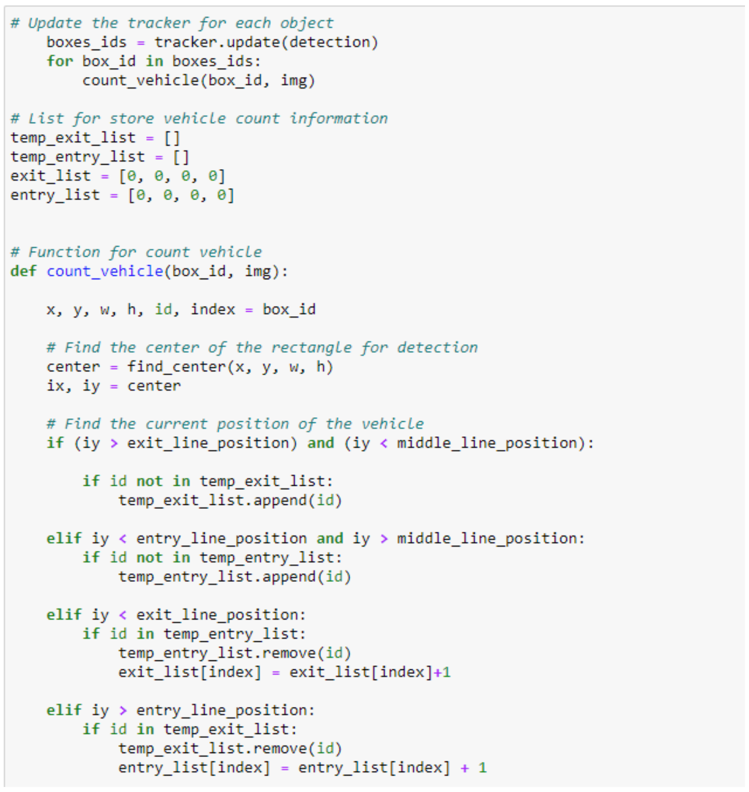


```python
# Function for finding the detected objects from the network output
detected_classNames = []
def postProcess(outputs,img):
    global detected_classNames
    height, width = img.shape[:2]
    boxes = []
    classIds = []
    confidence_scores = []
    detection = []
    for output in outputs:
        for det in output:
            scores = det[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if classId in required_class_index:
                if confidence > confThreshold:
                    # print(classId)
                    w,h = int(det[2]*width) , int(det[3]*height)
                    x,y = int((det[0]*width)-w/2) , int((det[1]*height)-h/2)
                    boxes.append([x,y,w,h])
                    classIds.append(classId)
                    confidence_scores.append(float(confidence))


# Apply Non-Max Suppression
    indices = cv2.dnn.NMSBoxes(boxes, confidence_scores, confThreshold, nmsThreshold)
    # print(classIds)
    if len(indices) > 0:
        for i in indices.flatten():
            x, y, w, h = boxes[i][0], boxes[i][1], boxes[i][2], boxes[i][3]
            # print(x,y,w,h)

            color = [int(c) for c in colors[classIds[i]]]
            name = classNames[classIds[i]]
            detected_classNames.append(name)
            # Draw classname and confidence score
            cv2.putText(img,f'{name.upper()} {int(confidence_scores[i]*100)}%',
                      (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

            # Draw bounding rectangle
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 1)
            detection.append([x, y, w, h, required_class_index.index(classIds[i])])
```

Figure No. 7.7: Creation of post processing module

7.5 Track and count all vehicles on the road

The number of vehicles that traversed the road can be determined with the use of a specialized function called *count_vehicle()* as created in Figure No. 7.8. Here, record the locations of the vehicles and the unique identifiers for each one. For exit vehicle counting, first determine if the item is between the exit crossing line and the middle crossing line, and then save the object's id in *exit_list* temporally. The reverse is true for automobiles travelling in the opposite direction. Then, look to see if the object has fallen below the exit line. If the object's id was detected above the entry line, it was treated as an entry route vehicle, and its class counter was incremented by 1.



```
# Update the tracker for each object
    boxes_ids = tracker.update(detection)
    for box_id in boxes_ids:
        count_vehicle(box_id, img)

# List for store vehicle count information
temp_exit_list = []
temp_entry_list = []
exit_list = [0, 0, 0, 0]
entry_list = [0, 0, 0, 0]


# Function for count vehicle
def count_vehicle(box_id, img):

    x, y, w, h, id, index = box_id

    # Find the center of the rectangle for detection
    center = find_center(x, y, w, h)
    ix, iy = center

    # Find the current position of the vehicle
    if (iy > exit_line_position) and (iy < middle_line_position):

        if id not in temp_exit_list:
            temp_exit_list.append(id)

    elif iy < entry_line_position and iy > middle_line_position:
        if id not in temp_entry_list:
            temp_entry_list.append(id)

    elif iy < exit_line_position:
        if id in temp_entry_list:
            temp_entry_list.remove(id)
            exit_list[index] = exit_list[index]+1

    elif iy > entry_line_position:
        if id in temp_exit_list:
            temp_exit_list.remove(id)
            entry_list[index] = entry_list[index] + 1
```

Figure No. 7.8: Tracking of all detected object

7.6 Vehicle detection, counting of entry / exit and available parking capacity

As employee’s car centroid moves from middle line to entry line, “Entry” field counter is updated and subsequently reduces the “Available” field value by -1 as in the Figure No. 7.9.

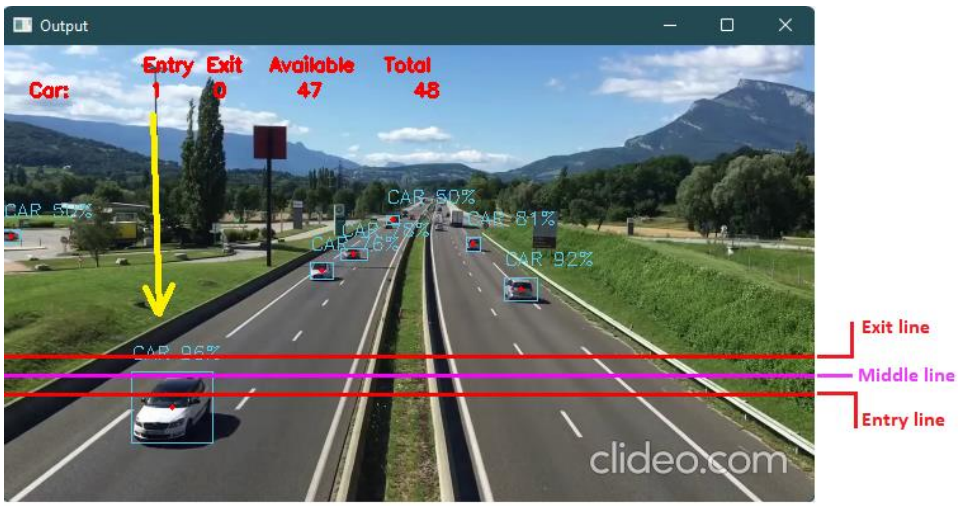


Figure No. 7.9: Vehicle entry count

As employee's car centroid moves from middle line to exit line, "Exit" field counter is updated and subsequently updating "Available" field value by 1 as in the Figure No. 7.10.

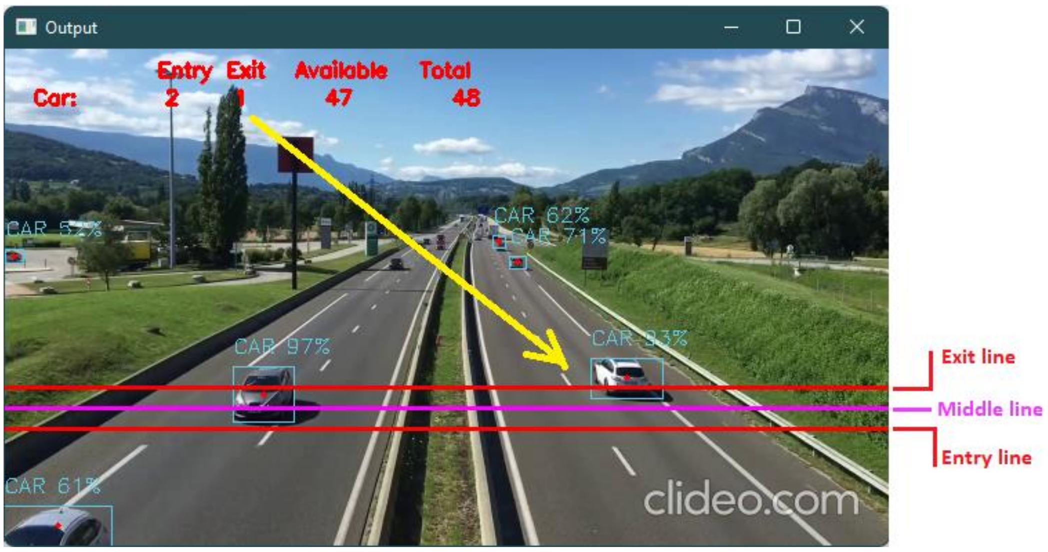


Figure No. 7.10: Vehicle exit count

Chapter 8: Testing and validation

The vehicle counting system developed in this work is tested using two different videos as described in previous section. Since proposed solution used a pretrained YOLOv4, so there is no training phase in this work. In the first testing scenario, system using video with 1280 x 720 resolution, 30 frame per second and totally has about 2460 frames and in the second scenario, same video with 15 frame per second was used. Both the video has 48 cars passing in enter and exit path each.

Differences between the actual number of vehicles and the number counted by the system are used to evaluate the system's efficiency. Model accuracy, as determined by using the Equation No. (1) and the resultant accuracy of the method is mentioned in the Table No. 8.1.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Actaul\ number\ of\ vehicle} \tag{1}$$

	Entry		Exit		Entry Accuracy	Exit Accuracy	Overall Accuracy
	Actual	Predicted	Actual	Predicted			
Video with 30 fps	48	48	48	48	100%	100%	100%
Video with 15 fps	48	48	48	45	100%	93.8%	96.9%
					Average Accuracy		98.45%

Table No. 8.1: Accuracy result of the model

Model output with 30 fps video:

The outcome of the result after running the model though video with 30 fps is presented in Figure No. 8.1.

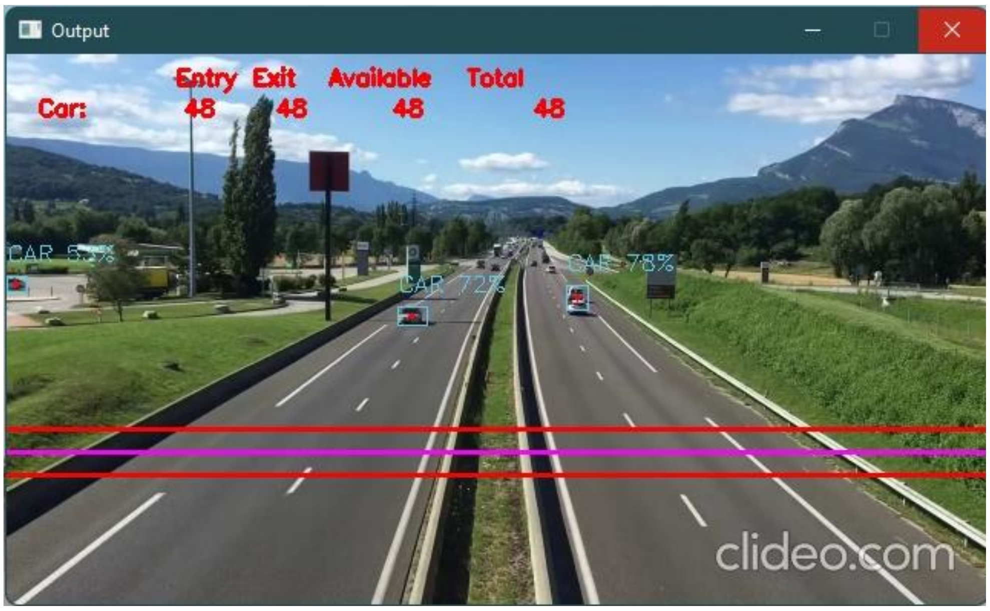


Figure No. 8.1: Model output with 30 fps video feed

Model output with 15 fps video:

The outcome of the result after running the model though video with 30 fps is presented in Figure No. 8.2.

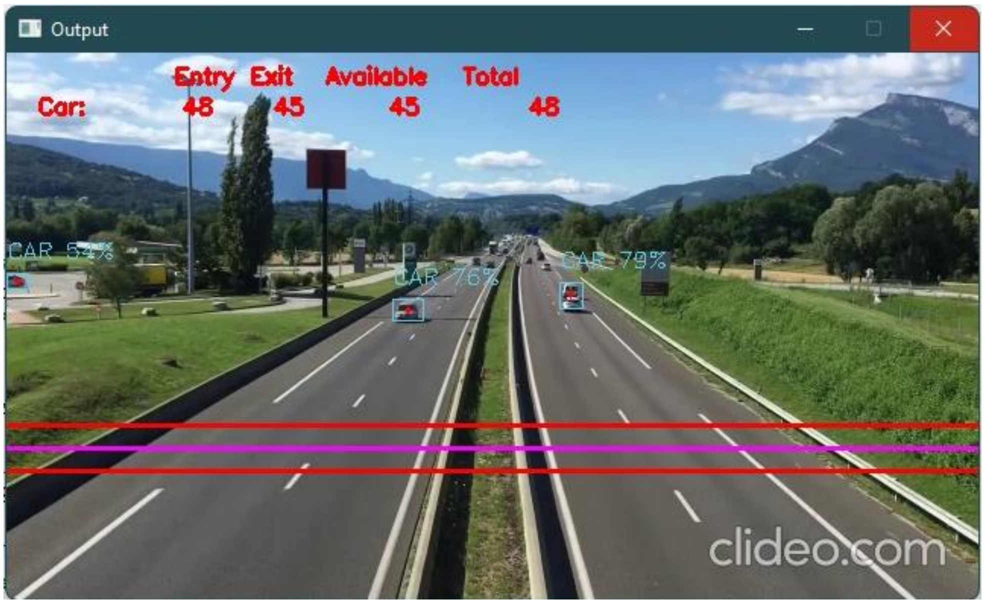


Figure No. 8.2: Model output with 15 fps video feed

Chapter 9: Analysis and Results

Application of the model is tested with two videos of different frame rate and accuracy of the modes are noted to understand the best fit model for a particular design in the previous chapter. Video with 30fps achieved 100% accuracy.

Entry and exit, detection and counting of vehicle with 30fps video of has done exceptionally well in overall detection and counting of the vehicle entering and exiting the premises. It is noted that both entry and exit accuracy achieved is 100%.

Entry and exit, detection and counting of vehicle with 15fps video of has done exceptionally well in detection and counting of the vehicle entering the premises and 6% times it fails to count the exit of vehicle. It is noted that overall entry and exit accuracy achieved is 93.8%.

It is noted that more frame rate means more information that video carries which is integral information, vital for DL model to process, track, and count. Hence from the experiment result it is evident that video feed or recording at 30 frame per second perform with 100% accuracy. Hence recommended to use separate 30 fps direct live video feed non-storage camera apart from regular storage-based surveillance camera to avoid investing on huge onsite storage.

Chapter 10: Conclusions and Future Scope

Smart parking solution which is developed by employing vehicle counting system at both entry and exit which process live or video feed has been developed using YOLOv4 for object detection, tracking by calculating Euclidean distance between centroid of a vehicle between frames and counting is simply executed by evaluating the distance between the vehicle's centroid to the border line. It successfully achieved the highest accuracy of 100% when using 30fps video feed as discussed in chapter 9, then following Table below. The frame rate of the video affects performance since it symbolises the information integrity of the data that the system processes. Overall, this work was successfully finished with positive outcome and accuracy of the model is presented in Table No. 10.1.

A dashboard with the status of available car parking can be placed at main entrance of each building campus along with the parking status of rest of the other buildings within campus so that it helps employees proceed to the building where parking space is available without having to shuttle across building to find the space predominantly during peak hour.

Selection of the frame rate can made based on the storage capacity or non-storage live feed can be used to for high accuracy and to avoid additional cost on storage.

	Entry		Exit		Entry Accuracy	Exit Accuracy	Overall Accuracy
	Actual	Predicted	Actual	Predicted			
Video with 30 fps	48	48	48	48	100%	100%	100%
Video with 15 fps	48	48	48	45	100%	93.8%	96.9%
					Average Accuracy		98.45%

Table No. 10.1: Accuracy result of the model

Smart Parking Solution for Multi Building Campus

Rajanna
Reva Academy of Corporate Excellence
Reva University
Bangalore, India
rajannak.ai01@race.reva.edu.in

Dr. Rashmi Agarwal
Reva Academy of Corporate Excellence
Reva University
Bangalore, India
rashmi.agarwal@race.reva.edu.in

Ravi Shukla
Reva Academy of Corporate Excellence
Reva University
Bangalore, India
ravishukla@race.reva.edu.in

***Abstract*— Locating parking space is becoming a serious problem as more and more employees prefer to switch to personal commute due to prevailing pandemic situation across globe in office campus. Particularly, where there is multiple building inside main campus, and it is quite challenging to keep track of car parking inside a given building. Without parking availability information, commuter must shuttle around builds to find parking space during peak hours, and this will be a loss of time as well as frustration to the commuter. There are few options companies follow like physically counting cars at certain interval and update the dashboard, install badge reader at entry and exit and people must punch card at both the places, sensors have been installed in certain parking lots, allowing management to track when cars enter and exit, install Radio Frequency Identification (RFID) and reader at entry and exit in order to track the capacity. All these methods will bring in burden on additional resource and high cost of installation of reader & necessary barricade systems respectively. Also, will add to regular maintenance of overall system. This project details DL implementation to create a vehicle counting system by using video from camera installed at the entry and exit path for vehicle. Pretrained model of YOLOv4 is used for object detection, vehicle tracking with Euclidean distance and counting. Since vehicle movement is considerably slow in pathway, performance of the model tested is 96.8% to 100% for varied frame rate of the video in parking capacity tracking.**

Keywords— Deep learning, Yolov4, Object detection, Vehicle counting, RFID, Parking

I. Introduction

It is challenging to keep track of available car parking space inside a given building. There are multiple buildings in main campus. Offices or buildings with separate entry and exit or single entry & exit. Post pandemic, employees prefer to use personal commute then public. Without parking availability information, employees must shuttle around builds to find parking space during peak hours, and this will be a loss of time as well as frustration experience to the commuter. In recent years, numerous proposals have been made for improved parking occupancy systems along with guidance. There are benefits and drawbacks to each of these modern systems. A method of determining whether or not a car is entering or exiting in a given space is essential to all of these systems. This mechanism could be anything from a straightforward ultrasonic sensor that detects a vehicle at a threshold distance to a sophisticated optical sensor that activates dependent on distance. The presence of a vehicle or other object is detected by these sensors. Much research has been conducted for traffic management applications based on image and video processing approaches. The analysis of traffic video data includes detection or recognition of vehicles, measurement of vehicle's speed, generation of tracking trajectory, counting of vehicles, congestion of traffic and collisions of vehicles. These applications have become popular recently due to the availability of low-cost cameras and embedded devices. Thus, the video data analytic is one of the prime research projects focuses in the computer vision and big-data area. This work proposes an easy-to-use system for counting vehicles based on the deep learning algorithm. A well-known pretrained YOLOv4 DL architecture known for its excellent accuracy in object detection and its moderate computation compared to other DL architectures. Vehicle tracking with Euclidean distance and counting entry and exit vehicles for accurate for parking capacity tracking.

II. Literature review

The counting of vehicles is often performed by splitting the scene into a moving foreground and a stationary background. Background subtraction (BS) and blob analysis [1], BS combined with Gaussian Mixture Model (GMM) based BS [2] [3], and particle filter-based tracker [3], [4] are used to achieve this goal. Frame averaging, the Gaussian mixture model, and principal component images are common methods for achieving BS, However, when dealing with traffic data, the background removal method has its limits. Partial occlusion in processed picture data causes vehicles to merge, and inaccurate bounding box predictions are made. Additionally, car shadows lead to erroneous detection. The results are enhanced by combining a Gaussian mixture model and an expectation maximization to create the background model. The moving vehicles are then retrieved from the background using subtraction. Morphological features and colour histograms are used to deal with the occlusions. One application of Computer Vison (CV) technology is in traffic monitoring systems, which can be used to keep tabs on things like vehicle speeds, parking lot occupancy rates, and traffic violations. The first step in every one of the aforementioned chores is finding where each car now resides. Therefore, an object detection method is so important. Pre-processing techniques like grayscale picture scaling, binarization of images, and background reduction [5] [6] or even edge detection is required by the traditional Machine Learning approach to accomplish this task. There are, of course, drawbacks to this method; for example, if the vehicle's shadow is present in the image, the detection may be less accurate.

Changes to the road surface, such as road maintenance, road damage, or any barriers on the road, can also cause inaccurate detection since they can disrupt the image subtraction process. Although it is computationally expensive and requires a considerable amount of data to train the networks, the Deep Learning (DL) methodology provides more adaptable performance without the requirement to pre-process the image and extract the feature using multiple methods. There are even more advanced DL architectures available, which have been trained with millions of data points, making the creation of CV systems much simpler. It is well-known that the Convolutional Neural Networks (CNN) architecture of Deep Learning (DL) performs exceptionally well in Computer Vision (CV), particularly in the tasks of object detection and categorization. Regional-based CNN (R-CNN), Fast R-CNN, Faster R-CNN, Region-based Fully CNN (R-FCN), Single Shot Detector (SSD), Mask R-CNN, YOLO, YOLOv2, YOLOv3, and YOLOv4 are only some of the CNN-based architectures used in CV for object recognition and categorization. For this experiment, YOLOv4 was chosen as the best model to utilise because of its high accuracy and fast computation time in real time compared to the other object identification methods.

III. Problem statement

Locating parking space is becoming a serious problem as more and more employees prefer to switch to personal commute due to prevailing pandemic situation across globe in office campus. Particularly, where there is multiple building main campus, and it is quite challenging to keep track of car parking inside a given building. Without parking availability information, employees must shuttle around builds to find parking space during peak hours, and this will be a loss of time as well as frustration to the commuter. There are few options companies follow like physically counting cars at certain interval and update the dashboard, install badge reader at entry and exit and people have to punch card at both the places, sensors have been installed in certain parking lots, allowing management to track when cars enter and exit., install Radio Frequency Identification (RFID) tag on each employee vehicle and reader at entry and exit in order to track the capacity. Very few or negligible companies have installed simple ultrasonic sensor at each parking space, and these are hardwired to the centralized system to detect parking occupancy. All these methods will bring in burden on additional resource and high cost of installation of reader & necessary barricade systems respectively. Also, will add to regular maintenance of overall system.

IV. Proposed work

This project proposes to use advancement in deep learning robust method You Only Look Once (YOLO) which treats the object detection as a regression problem to map pixels into bounding boxes with class probabilities. Moreover, it computes everything in a single evaluation. Euclidean distance methodology for object tracking and boundary line to count entry or exit of the vehicle in a single frame of the video to capture the parking availability in each building. The primary objective of this study is to implement the customized smart parking solution for multibuilding campus offices. Having multibuilding it is extremely important to let the employees know where the parking space is available especially during peaking hour. A dashboard at each building entrance encompassing overall occupancy and available status will greatly help the employee peacefully approach the building where parking is available. Secondary objective of the study helps in selection of optimal video mode for better accuracy for model for vehicle detect and counting.

V. Project methodology

The vehicle counting system which was built in this work has three main modules, i.e., Object Detection Module, Vehicle Tracking Module and Counting Module as given in Fig -1. The first module reads every single frame from the video and doing vehicle detection using YOLOv4 algorithm. This module results the location of every detected vehicle, i.e., the bounding box coordinates. The second module tracks vehicle by its centroid location between frames to frame using Euclidean distance by checking centroid distance between frames is within threshold, to ensure same object is being tracked and the third module counts the number of vehicles that crossed the road depending on the coordinates or position of the vehicles. So, the result of object detection module plays vital function in this system, since if the vehicle is not spotted, then it will not be counted. Video containing entry and exit of the vehicle from singe camera is used to demonstrate the proposed solution.

A. *Flowchart of proposed methodology*

The workflow for vehicle detection at entry and exit is presented in the Fig. 1. The workflow in the new approach is shown in Fig. 2. In the new approach, DL is used to compute the depth estimation approximation.

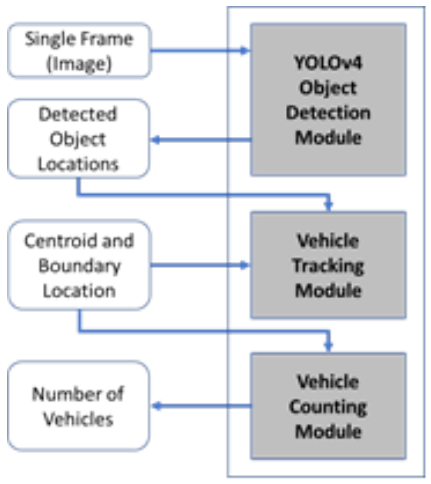

Single Frame (Image)
YOLOv4 Object Detection Module
Detected Object Locations
Centroid and Boundary Location
Vehicle Tracking Module
Number of Vehicles
Vehicle Counting Module

Fig. 1. General architecture of the vehicle counting system.

B. *Dataset*

Data for the proposed solution is implementation using video from surveillance camera having 81 second video with 1280 x 720 pixel having 30 frame per second frame rate and same video is converted to 15 frame per second frame rate for the study. This video has two-way traffic by which both entry and exit of vehicle data has been extracted to run through the model. In both the direction 48 cars will pass in enter and exit path each.

Specifically, video of different frame rate was taken because more the frame rate means higher storage capacity of storage for surveillance. Hence most of the companies record the video at lower frame per second to reduce the storage space and the cost associated with it. Snap of the video is shown in the Fig. 2.

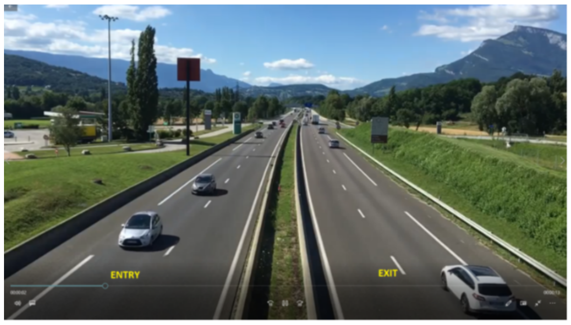


Fig. 2. Single image from a video

C. *Experimental setup*

The proposed solution implemented using OPENCV 4.5.2 and PYTHON 3.8.5 in a machine with Intel(R) Core i5-10210U, CPU of 2.10GHz processor with 16GB RAM. Even though this engine is not as fast as the GPU enabled system, but sufficient to run the deep learning.

D. *Module processing flow*

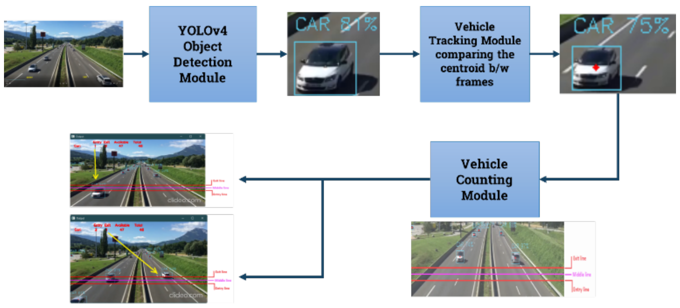

YOLOv4 Object Detection Module
CAR 81%
Vehicle Tracking Module comparing the centroid b/w frames
CAR 75%
Vehicle Counting Module

Fig. 3. Output flow from each module

VI. Result analysis

The vehicle counting system developed in this work is tested using two different videos as described in previous section. Since proposed solution used a pretrained YOLOv4, so there is no training phase in this work. In the first testing scenario, system using video with 1280 x 720 resolution, 30 frame per second and totally has about 2460 frames and in the second scenario, same video with 15 frame per second was used. Both the video has 48 cars passing in enter and exit path each. Differences between the actual number of vehicles and the number counted by the system are used to evaluate the system's efficiency. The resultant accuracy of the method is mentioned in the Table I.

TABLE I. Accuracy result of model

	Entry		**Exit**		**Entry Accuracy**	**Exit Accuracy**	**Overall Accuracy**
	Actual	***Predicted***	***Actual***	***Predicted***			
Video with 30 fps	48	48	48	48	100%	100%	100%
Video with 15 fps	48	48	48	45	100%	93.8%	96.9 %
Average Accuracy							98.45 %

VII. Conclusion and future work

Smart parking solution which is developed by employing vehicle counting system at both entry and exit which process live or video feed has been developed using YOLOv4 for object detection, tracking by calculating Euclidean distance between centroid of a vehicle between frames and counting is simply executed by evaluating the distance between the vehicle's centroid to the border line. It successfully achieved the highest accuracy of 100% when using 30fps video feed as discussed in previous section. The frame rate of the video affects performance since it symbolizes the information integrity of the data that the system processes. Overall, this work was successfully finished with positive outcome and accuracy of the model is presented in Table -1. A dashboard with the status of available car parking can be placed at main entrance of each building campus along with the parking status of rest of the other buildings within campus so that it helps employees proceed to the building where parking space is available without having to shuttle across building to find the space predominantly during peak hour. Selection of the frame rate can made based on the storage capacity or non-storage live feed can be used to for high accuracy and to avoid additional cost on storage. The proposed solution can further be extended to count other objects like motorcycle, bicycle...etc. With a higher-resolution camera, the same method can be applied to extract license plate information as well. Also, dashboard data can be made available to the employees over cloud or employee portal so that they can be well informed.

References

[1] T. H. Chen, Y. F. Lin, and T. Y. Chen, "Intelligent vehicle counting method based on blob analysis in traffic surveillance," Second Int. Conf. Innov. Comput. Inf. Control. ICICIC 2007, pp. 1–4, 2007, doi: 10.1109/ICICIC.2007.362.

[2] P. K. Bhaskar and S. P. Yong, "Image processing based vehicle detection and tracking method," 2014 Int. Conf. Comput. Inf. Sci. ICCOINS 2014 - A Conf. World Eng. Sci. Technol. Congr. ESTCON 2014 - Proc., 2014, doi: 10.1109/ICCOINS.2014.6868357.

[3] H. Jang, I. Won, and D. Jeong, "Automatic Vehicle Detection and Counting Algorithm," Int. J. Comput. Sci. Netw. Secur., vol. 14, no. 9, pp. 99–102, 2014.

[4] J. Quesada and P. Rodriguez, "Automatic vehicle counting method based on principal component pursuit background modeling," Proc. - Int. Conf. Image Process. ICIP, vol. 2016-Augus, pp. 3822–3826, 2016, doi: 10.1109/ICIP.2016.7533075.

[5] A. J. Kun and Z. Vámossy, "Traffic monitoring with computer vision," SAMI 2009 - 7th Int. Symp. Appl. Mach. Intell. Informatics, Proc., pp. 131–134, 2009, doi: 10.1109/SAMI.2009.4956624.

[6] Z. Iftikhar, P. Dissanayake, and P. Vial, "Computer vision based traffic monitoring system for multi-track freeways," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8589 LNAI, pp. 339–349, 2014, doi: 10.1007/978-3-319-09339-0_35.

[7] C. S. Asha and A. V. Narasimhadhan, “Vehicle Counting for Traffic Management System using YOLO and Correlation Filter,” 2018 IEEE Int. Conf. Electron. Comput. Commun. Technol. CONECCT 2018, no. March, 2018, doi: 10.1109/CONECCT.2018.8482380.

[8] M. Fachrie and U. T. Yogyakarta, “Model,” no. June, 2020, doi: 10.13140/RG.2.2.15026.56001.

The proposed solution can further be extended counter other objects like motorcycle, bicycle. ..etc. With a higher-resolution camera, the same method can be applied to extract license plate information as well. Also, dashboard data can be made available to the employees over cloud or employee portal so that they can be well informed.

Bibliography

[1] A. Kianpisheh, N. Mustaffa, P. Limtrairut, and P. Keikhosrokiani, "Smart Parking System (SPS) architecture using ultrasonic detector," *Int. J. Softw. Eng. its Appl.*, vol. 6, no. 3, pp. 51–58, 2012.

[2] B. Li and T. Lead, "Smart Applications !," pp. 18–20, 2011.

[3] G. Yan, S. Olariu, M. C. Weigle, and M. Abuelela, "SmartParking: A secure and intelligent parking system using NOTICE," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, no. April 2011, pp. 569–574, 2008, doi: 10.1109/ITSC.2008.4732702.

[4] S. Chaudhar, "Detecting Parking Spaces using Deep Learning Method: Solution for Parking in Smart Cities," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 7, no. 7, pp. 684–689, 2019, doi: 10.22214/ijraset.2019.7109.

[5] N. K. Chauhan and K. Singh, "A review on conventional machine learning vs deep learning," *2018 Int. Conf. Comput. Power Commun. Technol. GUCON 2018*, pp. 347–352, 2019, doi: 10.1109/GUCON.2018.8675097.

[6] N. O'Mahony *et al.*, "Deep Learning vs. Traditional Computer Vision," *Adv. Intell. Syst. Comput.*, vol. 943, no. Cv, pp. 128–144, 2020, doi: 10.1007/978-3-030-17795-9_10.

[7] A. Khanna and R. Anand, "IoT based smart parking system," *2016 Int. Conf. Internet Things Appl. IOTA 2016*, pp. 266–270, 2016, doi: 10.1109/IOTA.2016.7562735.

[8] C. Pornpanomchai, T. Liamsanguan, and V. Vannakosit, "Vehicle detection and counting from a video frame," *Proc. 2008 Int. Conf. Wavelet Anal. Pattern Recognition, ICWAPR*, vol. 1, pp. 356–361, 2008, doi: 10.1109/ICWAPR.2008.4635804.

[9] P. K. Bhaskar and S. P. Yong, "Image processing based vehicle detection and tracking method," *2014 Int. Conf. Comput. Inf. Sci. ICCOINS 2014 - A Conf. World Eng. Sci. Technol. Congr. ESTCON 2014 - Proc.*, 2014, doi: 10.1109/ICCOINS.2014.6868357.

[10] T. H. Chen, Y. F. Lin, and T. Y. Chen, "Intelligent vehicle counting

method based on blob analysis in traffic surveillance," *Second Int. Conf. Innov. Comput. Inf. Control. ICICIC 2007*, pp. 1–4, 2007, doi: 10.1109/ICICIC.2007.362.

[11] H. Jang, I. Won, and D. Jeong, "Automatic Vehicle Detection and Counting Algorithm," *Int. J. Comput. Sci. Netw. Secur.*, vol. 14, no. 9, pp. 99–102, 2014.

[12] V. Gomes, P. Barcellos, and J. Scharcanski, "Image-based approach for detecting vehicles in user-defined virtual inductive loops," *J. Electron. Imaging*, vol. 27, no. 03, p. 1, 2018, doi: 10.1117/1.jei.27.3.033026.

[13] J. Quesada and P. Rodriguez, "Automatic vehicle counting method based on principal component pursuit background modeling," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2016-Augus, pp. 3822–3826, 2016, doi: 10.1109/ICIP.2016.7533075.

[14] G. Salvi, "An automated nighttime vehicle counting and detection system for traffic surveillance," *Proc. - 2014 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2014*, vol. 1, pp. 131–136, 2014, doi: 10.1109/CSCI.2014.29.

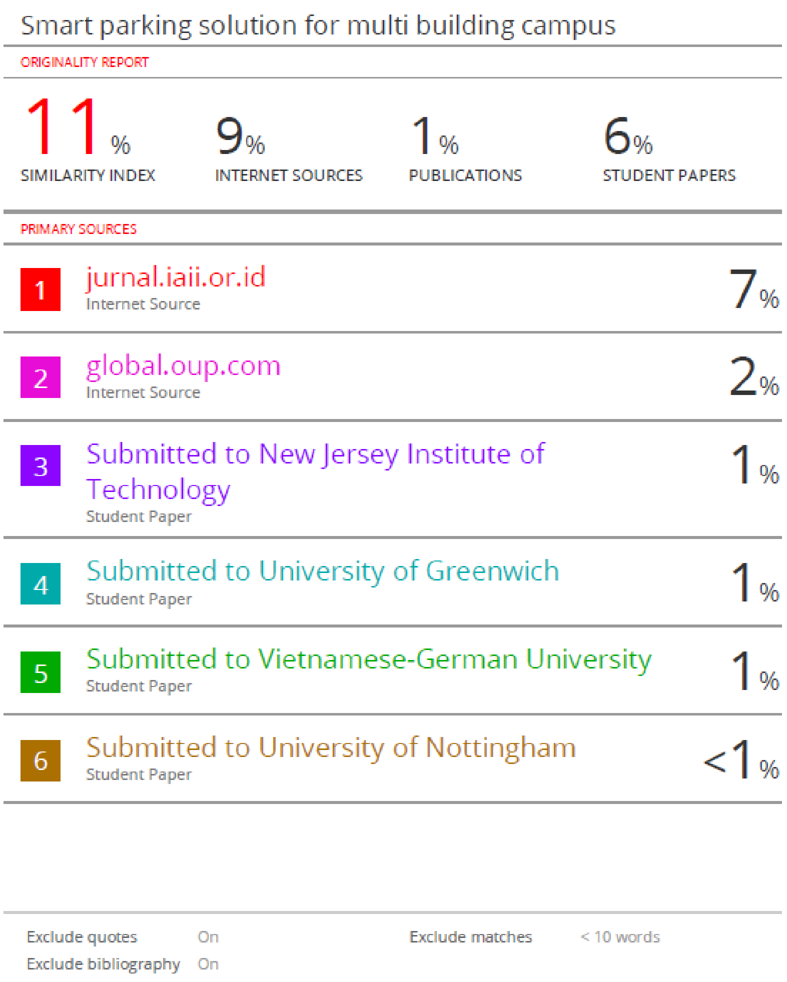
[15] T. Moranduzzo and F. Melgani, "Automatic car counting method for unmanned aerial vehicle images," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1635–1647, 2014, doi: 10.1109/TGRS.2013.2253108.

[16] A. J. Kun and Z. Vámossy, "Traffic monitoring with computer vision," *SAMI 2009 - 7th Int. Symp. Appl. Mach. Intell. Informatics, Proc.*, pp. 131–134, 2009, doi: 10.1109/SAMI.2009.4956624.

[17] Z. Iftikhar, P. Dissanayake, and P. Vial, "Computer vision based traffic monitoring system for multi-track freeways," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8589 LNAI, pp. 339–349, 2014, doi: 10.1007/978-3-319-09339-0_35.

[18] Patel, "済無No Title No Title No Title," 2019. https://github.com/charnkanit/Yolov5-Vehicle-Counting/blob/main/medium.mp4.

Appendix
Plagiarism Report



Smart parking solution for multi building campus

ORIGINALITY REPORT

11%	9%	1%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	jurnal.iaii.or.id Internet Source	7%
2	global.oup.com Internet Source	2%
3	Submitted to New Jersey Institute of Technology Student Paper	1%
4	Submitted to University of Greenwich Student Paper	1%
5	Submitted to Vietnamese-German University Student Paper	1%
6	Submitted to University of Nottingham Student Paper	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		