

# **Smart Vehicle Parking Management System using Image Processing**

Rahul Sharma

2/01/2022

## **1. Abstract**

The term parking management system usually refers to the custom-built hardware intensive systems installed in building and malls. However, there are many places where such expensive solutions cannot be installed due to various reasons, like cost and urgent/temporary setup requirements. This project focuses on developing a parking management system based on image processing to detect vacant parking slot in an area where automated systems are not installed. Camera images of the parking area are subjected to image processing algorithm which marks virtual slots in the area and extracts occupancy information to guide the incoming drivers about availability and position of vacant spaces. The application consists of two interfaces: one for the guidance of the incoming drivers and the other one for the administrator. The later interface also informs the administrator if a car is not parked properly in the virtual slot. This parking system would reduce the stress and time wastage associated with car parking and would make the management of such areas less costly.

## **2. Problem Statement**

The problem statement is to solve the problem of parking. Car parking is a major problem in urban areas in both developed and developing countries. This imbalance is partially due to ineffective land use planning and miscalculations of space requirements during first stages of planning. Shortage of parking space, high parking tariffs, and traffic congestion due to visitors in search for a parking place are only a few examples of everyday parking problems. There is a massive increase in the number of cars that are on the road in the past few years and the number is still increasing with increase in the number of cars on the road the issue of parking one's car is also increasing. Problem is increasing at such an extent that almost 40% of the roads are being used for parking in normal working days. People from the town like Delhi, Mumbai, Chandigarh, Noida and many more spends over 80 hours a year in search for parking space. Such towns also come under as one the of the most busiest road in the world. It's an effort to solve this problem by introducing a database system which would notify about the parking space and would allow for optimum place for the vehicle and the charge that would be taken. This would allow for the parking space to be filled rather than a vehicle searching long roads for the space. This would allow the management to notify if the parking space is filled so that no other car is allowed before a car leaves the spot.

### **2.1 Market/Customer/Business need Assessment**

If I talk about the parking problem then, it is true to state that it is the most underdog problem which is never discussed but it is a big deal. People going to public places like shopping mart, markets, malls and numerous places, no matter which area whether it's an Urban or rural locality. People do tackle such problems in their day-to-day life. Such problems can give rise to traffic jams which can lead to various problems like consumption of time, accidents and various problems. So, by using this solution, we can aim to modernise the parking system and can prevent the traffic jams.

### **3. Target Specification**

The proposed system will provide the parking owner with the easiness on viewing his respective parking area digitally and will easily be able to verify with the availability of the parking space. It will provide the information of whether the parking space is free or not. This will help in preventing the misleading of confusion regarding to the availability of parking space. It will also detect whether a car is parked perfectly or not. Also creating a certain app for this can help the customers in finding parking spaces near them and can also verify whether space is available or not. This can also help in keeping an eye on the vehicle. Overall, this can contribute to resolving the traffic issue.

### **4. Methodology**

This project, vehicle parking management system using image processing aims to create a better environment for a vision-based vacancy parking area detection, providing a modern and innovative solution for temporary parking places. For example, dust ground, cemented flooring where no specific parking systems are used. The prime objective is to have maximum number of cars which can be parked in an organized manner into the temporary lot. This project's aim is to detect and recognize the real time vacant parking space.

It comprises of a camera mounted on roof top of any nearby building or some supporting pole at certain angle where it covers the maximum area of parking lot which is being used for taking the input. The images obtained from the live stream are then fed to the processing module, which detects the region of interest (ROI) consisting of the area to be covered for parking spaces. A car detection Module is used to detect the cars within ROI using Neural Network. This module tracks and detects the parking space in an image. Python Programming language is used to train the models and do processing.

#### **4.1. ROI Detection**

The camera has been mounted on roof top of a building from where the parking lot is completely visible. First challenge is to detect the ROI in the image obtained from camera. For this purpose, markers are placed on the image corners using distinguishing colour. All images are resized to consistent dimensions and are then passed through two convoluting filters. The convoluted image is then made to pass through Gaussian blur, where the image is made slightly blurred. This process converts the original image to filtered image. The smooth and unique colour of markers is then detected using masking technique, based on the range of pixel values in the HSV colour space. In order to find out the coordinates of the mask, contours are located, which can be defined as the line of pixels whose values are same or in other word the edges in image are found. Since the camera is mounted on to roof top at some Angle, the ROI obtained is not a square, so the perspective of the image needs to be changed, to make it square because the car near to the camera is of different size as compared to the car placed far to the camera. Once ROI has been marked properly, the image data is labelled for supervised machine learning algorithm; labelled data set is required so that the machine can easily understand the input patterns. To train the model to detect the car annotation of the data set is needed. Till now the data has been collected, recreated, filtered and annotated, now it is ready for training the neural network.

## **4.2. COLLECTING THE DATA**

Collecting the data means gathering the image in order to train them. The image can be easily collected by using

Google images:

- ❖ There you can download various images according to your project. But the only main clause of using it is that you cannot download the multiple images until and unless you add multi-image downloader in chrome extension.

OID:

- ❖ Open Images Dataset is a free, open-source set of image data that can be downloaded and can be utilized to build a machine learning models. You can visit to the site and can explore various categories of images. It offers both bounding boxes and images for more than 600 different categories of objects, simplifying the process of getting started. OID has made the work easy to gather the data according to user's requirement. It provides with facilities of downloading multiple images either with bounded boxes or normal images.

## **4.3. DATA ANNOTATION**

Data annotation is the process of selecting a portion of an image and assigning a label to that region. For image classification, annotation is done by providing label to the entire image, instead for a specific region. Annotated data act as the input for model training via supervised learning. With various annotation done on the image, the machine can characterize the images, and learns to detect the objects of interest, classifying images etc., it completely depends on the type of model being trained. Examples for annotation are drawing bounding boxes or 3D cuboid boxes and assigning labels to objects for object detection and many more. Data annotation can be done by using the annotation tools. These tools bring the quickness and accuracy in order to build the ground truth for the computer vision models. There are various tools for annotation, some of them are as follows:

- LabelImg
- Lionbridge AI
- TrainingData.io
- Spare5

There are many more annotation tools but among them LabelImg is more commonly used.

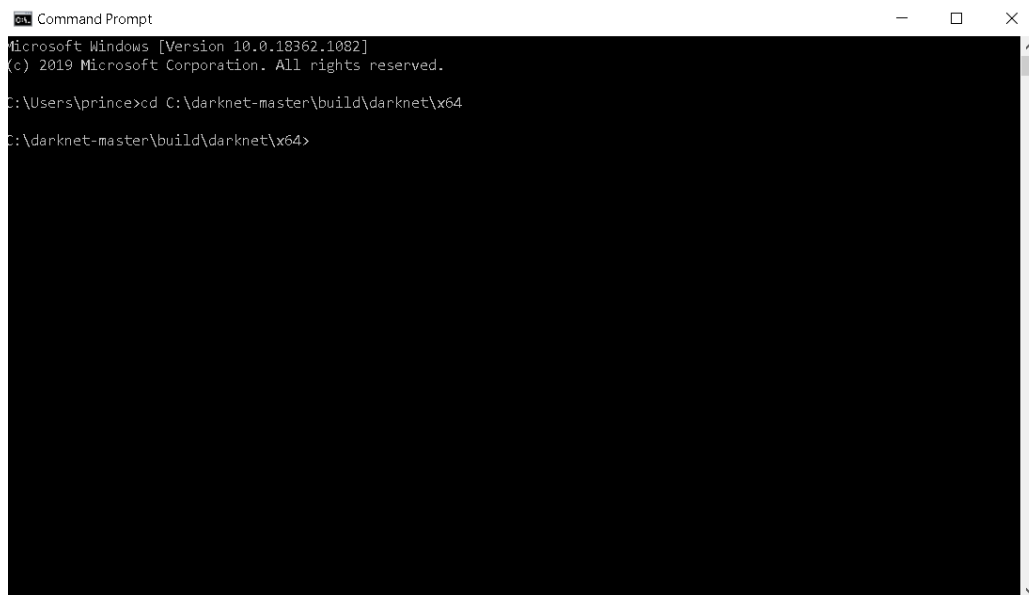
## **4.5. TRAINING THE MODEL**

After the data are collected and annotated, they can be utilized as input for model training. The data are then fed into the DNN, which then helps in making the prediction like a label for image classification, labels and bounding boxes for object detection, label maps for image segmentation, and keypoint sets for landmark detection. The model compares the prediction to the annotations and adjusts produce better predictions. One can create its very own computer vision model for the application, one can start from scratch and build its own model architecture or can use any existing one. Before training the model for computer vision, it is necessary to train the annotation. The best way to train your annotation is by using Darknet. It is very popularly known for its training the annotation.

### **4.5.1 Darknet**

Darknet is an open-source neural network framework written in C and CUDA. It is basically used for training the annotation in a very simple manner. The speed of training the images is much faster than TensorFlow. In order to use Darknet you need to download the specific required files and then you can use it. It is operated by using command prompt. We can also use it in both PowerShell as well as google colab. Below is the demonstration of how the darknet works. I have worked in windows platform with the help of command prompt to train my system.

#### **How it works**



```
Command Prompt
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\prince>cd C:\darknet-master\build\darknet\x64
C:\darknet-master\build\darknet\x64>
```

Since darknet is operated by using command prompt. So, very first need to enter the code for locating the darknet.exe or darknet\_no\_gpu.exe file. These files are firstly created with the help of visual studio. After generating the file then you can utilise to train your annotation.

Now once you done with that, you now can utilise the darknet. In order to train your annotation, you just simply need to write code line: (**darknet\_no\_gpu.exe detect cfg/yolov3.cfg yolov3.weights**) as an output it will start training images as shown below.

```

Select Command Prompt - darknet_no_gpu.exe detect cfg/yolov3.cfg yolov3.weights
Microsoft Windows [Version 10.0.18362.1002]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\prince>cd C:\darknet-master\build\darknet\x64

C:\darknet-master\build\darknet\x64>darknet_no_gpu.exe detect cfg/yolov3.cfg yolov3.weights
GPU isn't used
Used AVX
Used FMA & AVX2
OpenCV version: 3.3.0
mini_batch = 1, batch = 1, time_steps = 1, train = 0
layer  filters  size/stride/dill  input  output
0 conv  32      3 x 3 / 1        416 x 416 x 3 -> 416 x 416 x 32 0.299 BF
1 conv  64      3 x 3 / 2        416 x 416 x 32 -> 208 x 208 x 64 1.595 BF
2 conv  32      1 x 1 / 1        208 x 208 x 64 -> 208 x 208 x 32 0.177 BF
3 conv  64      3 x 3 / 1        208 x 208 x 32 -> 208 x 208 x 64 1.595 BF
4 Shortcut Layer: 1, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.001 BF
5 conv  128     3 x 3 / 2        208 x 208 x 64 -> 104 x 104 x 128 1.595 BF
6 conv  64      1 x 1 / 1        104 x 104 x 128 -> 104 x 104 x 64 0.177 BF
7 conv  128     3 x 3 / 1        104 x 104 x 64 -> 104 x 104 x 128 1.595 BF
8 Shortcut Layer: 5, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF
9 conv  64      1 x 1 / 1        104 x 104 x 128 -> 104 x 104 x 64 0.177 BF
10 conv 128     3 x 3 / 1        104 x 104 x 64 -> 104 x 104 x 128 1.595 BF
11 Shortcut Layer: 8, wt = 0, wn = 0, outputs: 104 x 104 x 128 0.001 BF
12 conv 256     3 x 3 / 2        104 x 104 x 128 -> 52 x 52 x 256 1.595 BF
13 conv 128     1 x 1 / 1        52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
14 conv 256     3 x 3 / 1        52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
15 Shortcut Layer: 12, wt = 0, wn = 0, outputs: 52 x 52 x 256 0.001 BF
16 conv 128     1 x 1 / 1        52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
17 conv 256     3 x 3 / 1        52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

```

After training the annotation, you can check your trained annotation model by entering the path of the image as after training is completed. It itself puts an option of checking progress by entering the path of image as shown below.

```

Command Prompt - darknet_no_gpu.exe detect cfg/yolov3.cfg yolov3.weights
101 conv 128     1 x 1 / 1        52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv 256     3 x 3 / 1        52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv 128     1 x 1 / 1        52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv 256     3 x 3 / 1        52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv 255     1 x 1 / 1        52 x 52 x 256 -> 52 x 52 x 255 0.353 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_xy: 1.00
total BFLOPS 65.879
avg_outputs = 532444
Loading weights from yolov3.weights...
Done! Loaded 107 layers from weights-file
Enter Image Path: C:\Users\prince\Desktop\cars\unnamed(1).jpg
Cannot load image C:\Users\prince\Desktop\cars\unnamed(1).jpg
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
C:\Users\prince\Desktop\cars\unnamed(1).jpg: Predicted in 2528.873000 milli-seconds.
Enter Image Path: C:\Users\prince\Desktop\cars\parking_story.jpg
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
C:\Users\prince\Desktop\cars\parking_story.jpg: Predicted in 2656.099000 milli-seconds.
car: 99%
car: 99%
car: 100%
car: 100%
car: 99%
car: 73%

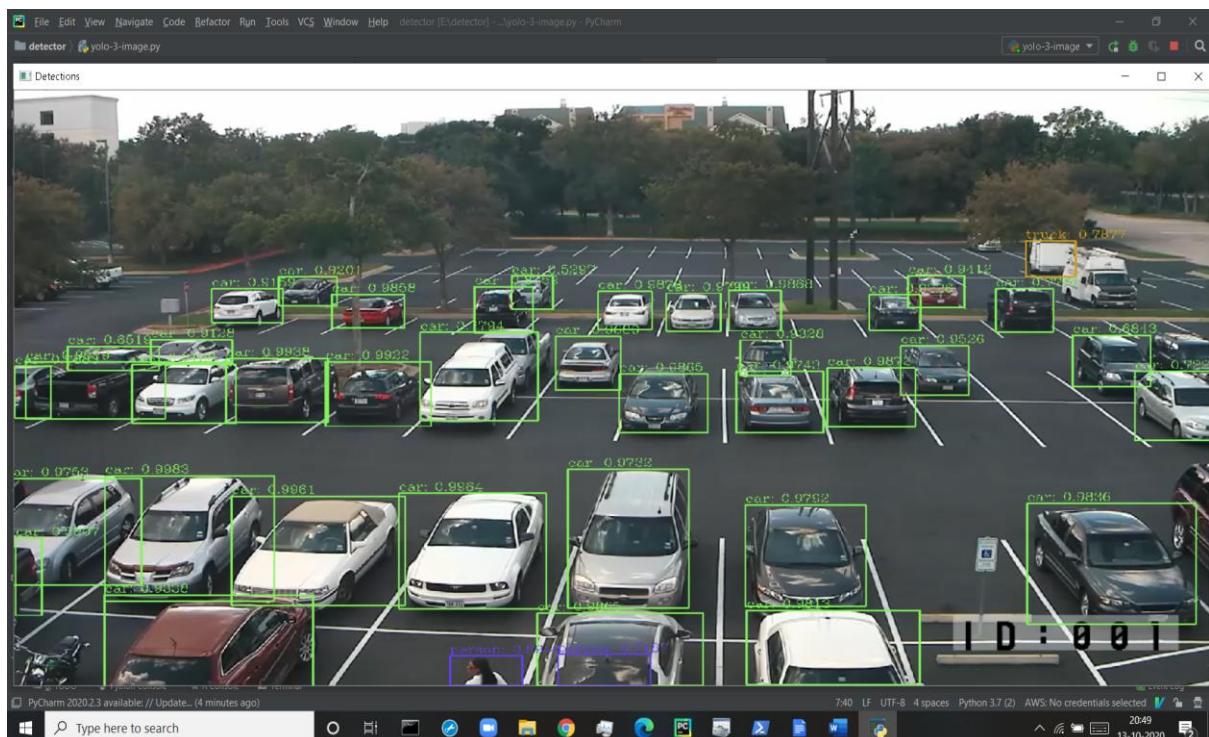
```

Once you enter the path of image and the image name. In the output you will get the image with bounded box and will be able to classify the class you trained in annotation. Below this represents the accuracy rate of our annotated trained model with classes it has able to classify. After using the command, the image will look like the below.



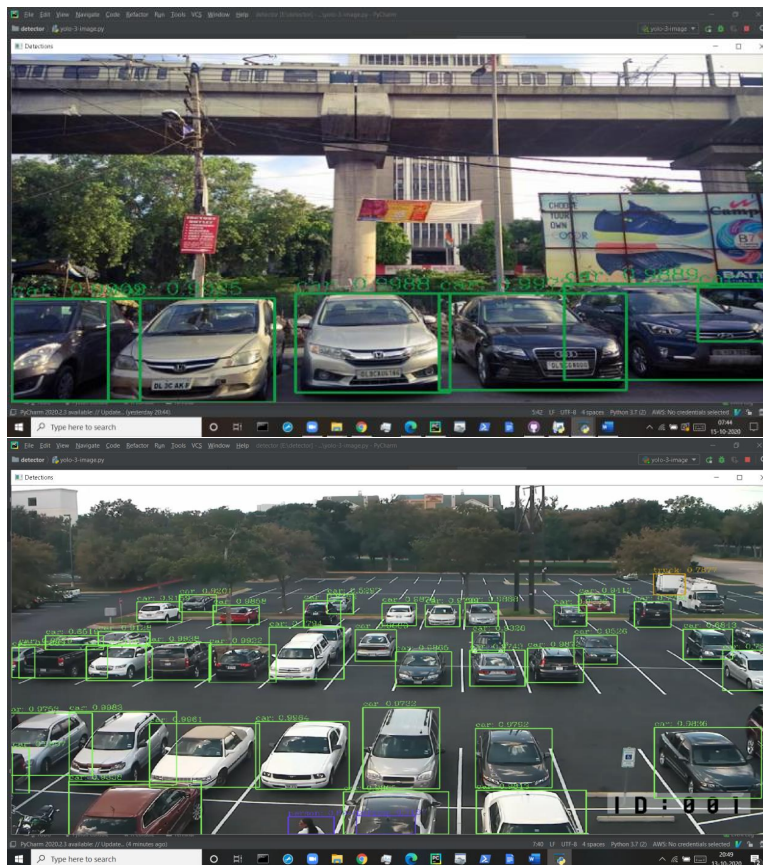
## 4.6. Car Detection

Once the model being trained the weights are being saved in H5 format that can be used for detecting a car in an image. The prepared images are fed to the neural network. The chosen model architecture for training is Mask RCNN. The resulting model detects the boundary of every car. While defining the architecture to load the weights, the confidence value is set to 0.9 that means an object which have 90% confidence to be a car is detected while objects having lesser confidence values are rejected. Below is the image of machine identifying the vehicles parking.

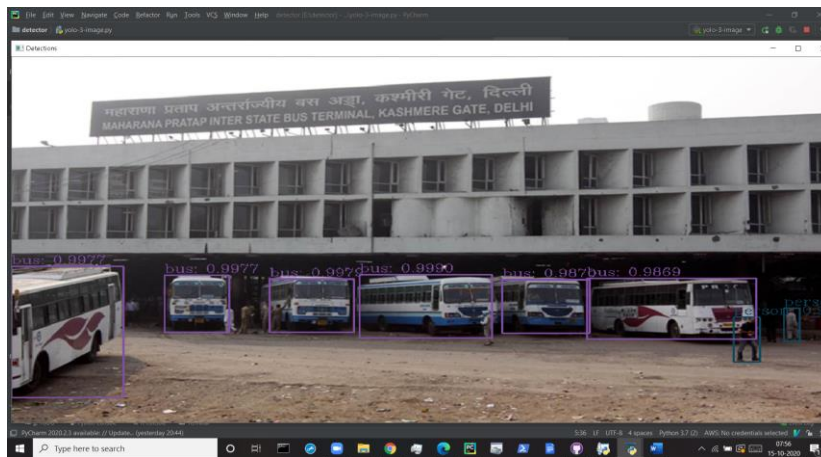


## 5. Current Progress

Till now work is on progress, the system can identify what vehicle is standing on the parking area. Like the images below. The blank one in the image is classified as the free space. But still lots of upgradation is required. I am planning to make it more detailed. I have planned to utilize the edge detection technique to show the parking space in detailed. With the help of edge detection, we will be able to identify whether the car is parked perfectly or not. After all of that I am having a vision to develop an app where a user can monitor for the parking space availability and an admin can keep a check on overall parking space occupied as well as parking space available. Below are some of glimpse of system working. But still trying to overcome to some problems.







## **6. Conclusion**

The Final Version of this Project is a mobile Application, offering a start-up product in management sector that aims to address the parking difficulty issues at some mega-events where vehicles must be parked in temporary parking area. The vision-based parking management system features to have maximum parking within the ROI and to facilitate the user with the best. The user can have a real-time parking lot update in order to see if there any vacant space available to park the car or not. Since most people are in hurry and park the car in the wrong way. In order to track it out, we have an admin application where the admin can check into the system for any wrong parking, the total number of vacant spaces, the number of correctly parked cars, and other details.

## **7. References**

[www.google.com](http://www.google.com)

[www.parknsecure.com](http://www.parknsecure.com)

<https://blog.getmyparking.com>

<https://www.ijsr.net>

<https://pjreddie.com/darknet>