AI ENABLE CAR PARKING USING OPENCV

A PROJECT REPORT

Submitted by

G.AJAY KUMAR (SBAP0004715)

G.SATYANARAYANA (SBAP0004719)

G.ABHINASH (SBAP0004717)

D.VAMSHI (SBAP0004713)

D.RAVI (SBAP0004714)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRICAL & ELECTRONICS ENGINEERING

AVANTHI INSTITUTE OF ENGINEERING ANDTECHNOLOGY

At chirukupally ,near tagarapuvalasa ,531162

**INDEX**

* INTRODUCTION ABOUT THE PROJECT
* Problem statement
* Identify the needs
* Raw Data ( information)
* According to other people who worked on the same project type
* Marketing requirements
* Research Survey or Literature Review This section first provides some important background in Section
* Research Survey or Literature Review The work in PARKSMARTER
* The work in AI Smart PARKING
* The work in Camera based outdoor parking guidance system
* The work in Camera Based PGS
* The work in Outdoor camera
* Build a simple smart parking project using python and OpenCV
* Overview
* Time to take action
* To infinity and beyond

Introduction

Car park owners and drivers have one thing related: they want parking operations to be easy, secure, and efficient. However, in practice, it can

be hard to find a free space after entering a car park. As a lot of drivers will know, this is very frustrating! We offer a way to guide drivers to parking their cars quickly, with products that can be too easy to be combined and integrated for a simple and efficient solution. Streamlining parking, both at Entrance-Exit and through the car park, make the parking much more efficient, speeding up the process and potentially freeing up spaces more quickly for further occupation. This means taking the maximum profits on spaces. It also provides a level of security, giving peace of mind to drivers and operators alike.

**Problem statement**:

This chapter first performs a need analysis and presents customer requirements in Section. It is followed by a research survey in Section . Finally, the need and objective statements are given in Section

**Identify the needs**:

The need analysis is performed in the following four steps:

.**Raw Data ( information** )

• Interviews with some students and professors in the university:

▪ According to students, they take 15-30 minutes to find a free parking space, and when they find one it will be far from campus and which causes they to get late to lectures, and they wish to find some screens or smart apps to help to guide them to the nearest free parking space and save some time.

▪ According to professors they find traffic before lectures time on the way to the campus, which causes accidents sometimes and they get stuck, they hope someone finds a solution to manage the traffic in the parking spaces.

**• According to other people who worked on the same project type :**

▪ Free space indicator Indoor Parking Guidance Cameras/ Outdoor Parking Guidance Camera detects a vehicle in space and shows its availability status automatically by colored lights, so drivers can easily see where to park. For example, it turns red for occupied and green for free. Different colors can also be used to categorize spaces, for instance for disabled parking.

▪ Free space count The number of available spaces can be displayed, for example at the entrance to a car park on an LED Display. This uses count information using ANPR data from entrance/exit cameras or Indoor Parking Guidance Cameras/Outdoor Parking Guidance Camera, and updates immediately when space fills, or becomes free. It informs a driver right from the start if there’s space inside the car park.

**Marketing requirements:**

1.The system should have high-quality cameras and radars

2. The system should have a good app

3. The system should be easy to operate

4. The system should be connected to Wi-Fi

5. The system should work all the time

6. The system should have an AI smart system

7. The system should be easy to fix or install

8. The system should have fast recognition for parking space

9. The system should be connected to screens

10. The maintenance time should be within 2-3 days maximum

11. The system should have night vision

12. The system should guide you to the closest free parking space according to your location

13. The system should have multiple cameras and radars

14. The system should recognize all types of vehicles 15. There should have a backup system

16. The system should have a small size

17. The system should have a reasonable cost

18. The system should have memory

19. The system should have an auto error detection

20. The system should be connected to GDOT to detect any traffic violation

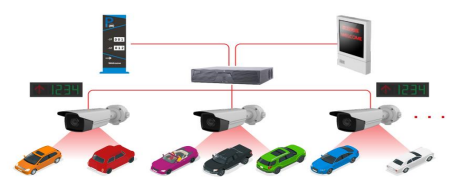
**Research Survey or Literature Review This section first provides some important background in Section**.

A comprehensive research survey is then provided in Section .

**Background**

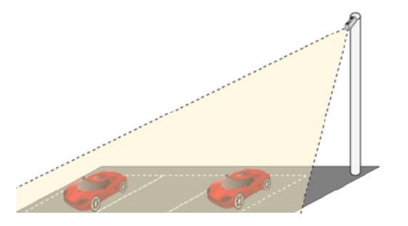
We chose this project or this problem because we all waste a lot of resources and time in parking spaces looking for a free spot for our vehicles, In our project, we provide a smart and fast way to find a free parking space and guide the driver to the nearest free spot according to his location whether he uses the LEDs screens or the App using GPS, The system use AI and multiple cameras and radars/lidars to detect the parking spaces to know if the spot is free to use or occupied, and also to save videos to the memory so it can be reached if needed in accidents or any other reason to save the rights of all the drivers, also, the system is connected to GDOT ( Government Department of Traffic ) to detect any traffic violation, so no one will take more than one parking space and no one will block any other cars in the parking spaces so the driver doesn’t need to worry about his car at all.

**Research Survey or Literature Review The work in**  **PARKSMARTER**

PLUS provides a free space indicator that detects if the space is free to use or occupied, also it has a free space count that shows the free spaces on screens to the drivers, and it has parking guidance so it helps the drivers to the nearest free spot 

**The work in AI Smart PARKING**

provides a high-resolution camera that automatically detectsfree or occupied parking spots throw advanced algorithms based on deep neural networks and sends the state of the parking spot through the application, also it can analyze crowded days in parking places and uncrowded days, also it's measures the time of stay in the parking spot.



**The work in Camera based outdoor parking guidance system** provides a single camera that can monitor 100 or more spaces and detection of poorly parked vehicles, The system can fit in large and small car parks, and also has LED screens to tell the driver where is the nearest parking available.



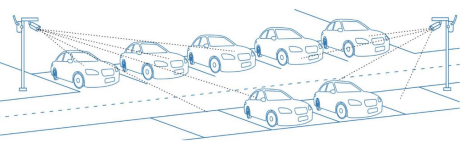
**The work in Camera Based PGS**

provides 24/7 video recording for the six parking spots which helps to trace theft, and accidents within the car park, also it can count the number of people in the car park, also it has an auto-detection of blacklisted vehicles, and the system gives you alert if the parking zone is crowded.



**The work in Outdoor camera**

provides AI deep learning algorithm which uses streams from CCTV cameras which make the cost less, also it can be installed in a second, also it has great visibility to any lightning, and can help customers to find empty parking places through LEDs sign or mobile app.



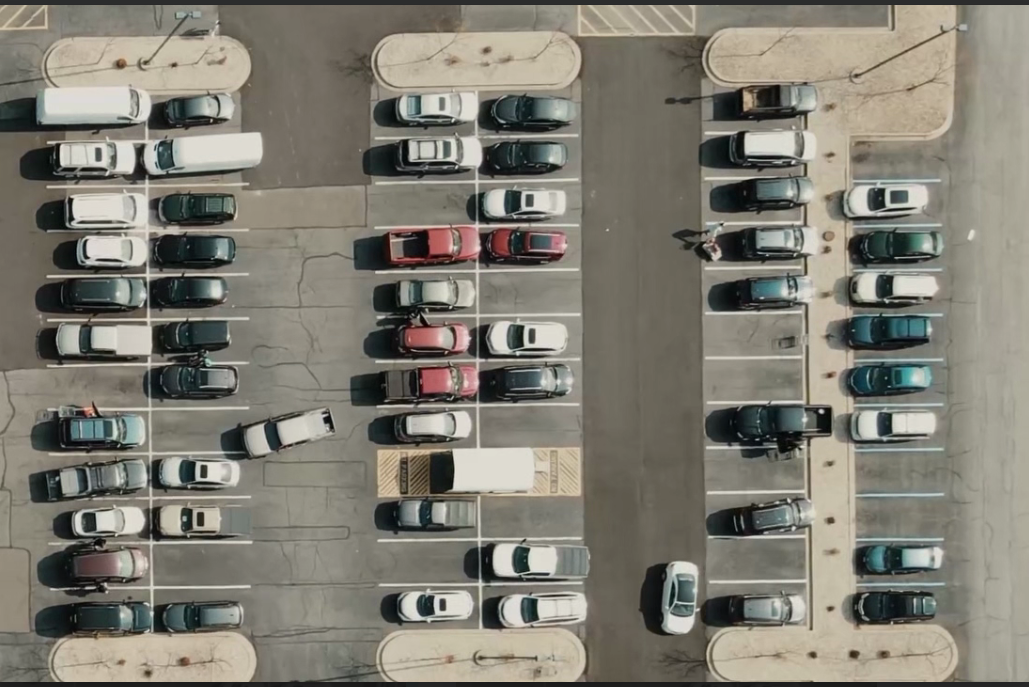
Build a simple smart parking project using python and OpenCV

In this project we use some python packages and python IDE such as pychram,anaconda ,visual studio code

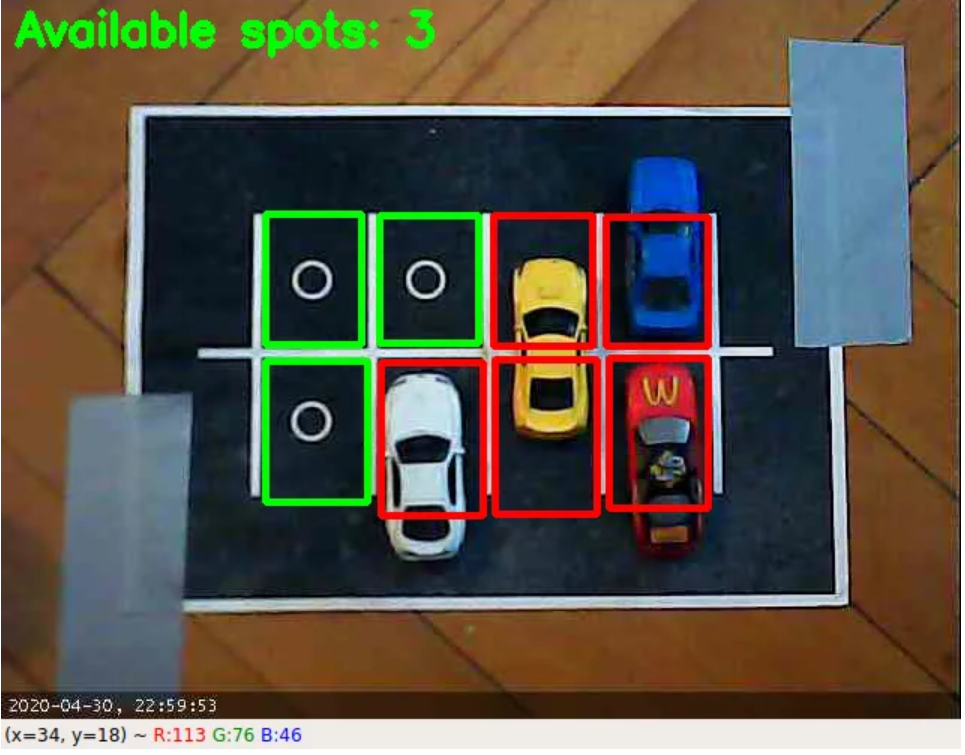
**Packages :**

* **Flask(flask**
* **Opencv(cv2)**
* **Regular expression(re)**
* **IBM Data base2(idm\_db)**
* **Numpy**
* **Pickle**
* **Cvzone**

**This input image our project :**

****

**Output of this project:**

****

## 

## **Backstory**

One of the most annoying things happening in our era is looking for a parking spot and not being able to find one, especially when you had to be somewhere five minutes ago and you are currently looking for a parking spot for 20 minutes now.

However, any problem must have a solution, or more solutions, based on complexity and efficiency. There are many solutions for smart parking systems out there, including deep learning implementations, weight sensors, light sensors and all of that science fiction stuff that surrounds the world these days.

## **Overview**

The concept behind this solution is quite simple. It is composed of two scripts with the following roles:

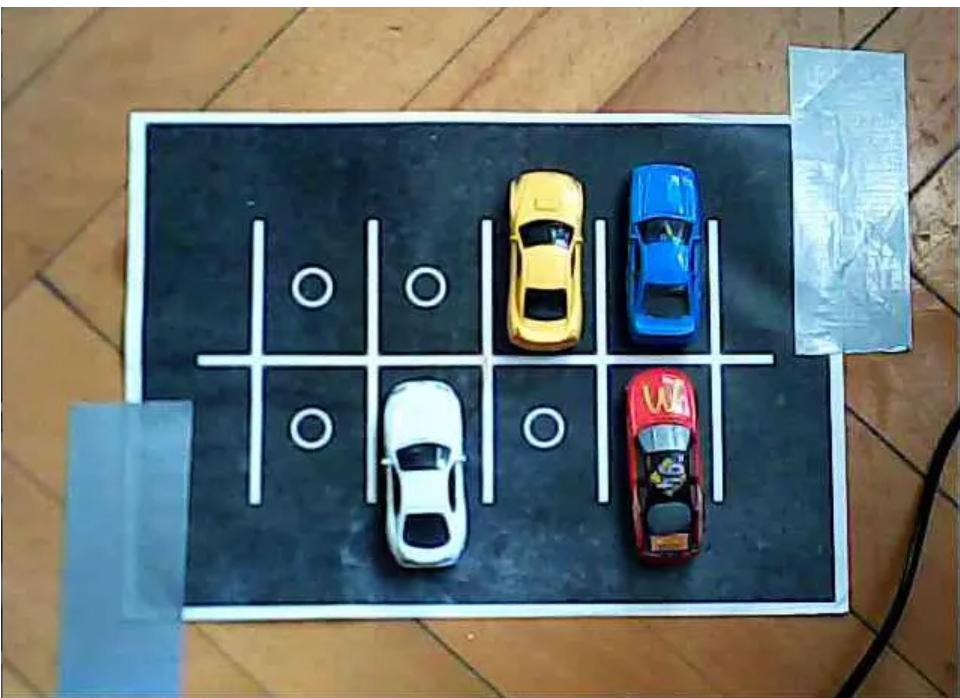
1. Select the coordinates of the parking spaces and save them into a file.
2. Get the coordinates from the file and decide if the spot is available or not.

The reason for splitting this solution in two scripts is strictly related to avoiding to select the spots every time you want to see if there are any available spots especially if used the same location as before.

To keep this as simple as possible, from now on I will refer to these scripts as **selector** and **detector**.

## **Time to take action**

First of all we will need to setup a parking lot camera. In my case, as I don’t have any parking lots that I can see from my windows I choose to use my old car toys and a printed paper to play with. Also I set up a webcam right above the parking lot to get a good image, so the image we are working on looks like this:

**To infinity and beyond**

What is happening inside our loop it’s actually a bit more complicated that just calling our build functions.

First we initialize the number of available spaces to 0 to prevent the addition of numbers for every frame.

Then we start two streams to show the real image and the image that is processed. This helps for a better understanding of how this script works and how the image is processed.

After that we need to get the values of the parameters set in the **parameters** GUI we created, each iteration. This is done with the cv2.getTrackbarPos function.

Now the most important part takes places, the appliance of **drawRectangle**function to all the coordinates taken by **Selector** script.

What’s left to do now is to write the number of available spots on the resulted image, display the **Canny**function result and, obviously, a well known way to stop our loop.

Voila! we have a smart parking project now!

