# Parking Slot availability status in Smart Campus

Ayush Mangukia - 191IT211
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:ayushmangukia.191it211@nitk.edu.in

Mohammed Ibrahim - 191IT230
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:mibrahim.191it230@nitk.edu.in

Soorya Golamudi - 191IT252
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: sooryahyma.191it252@nitk.edu.in

***Abstract*—Locating a non occupied parking slot in metropolitan areas using image processing techniques.**

**The global number of vehicles is around 1.35 billion and does not seem to go down any time soon. We are focusing on the parking obstacles; the parking lots cannot keep up with the number of automobiles and hence need to adapt. We are using low maintenance battery-powered cameras fixed around the parking lot to acquire the images, which are then processed in OpenCV. The python code then detects all the possible combinations of edges in the area that can serve as parking slots. To determine if they are empty or not, the code then creates a 'differential image.' The differential image is taken by the color difference of each pixel in two different image inputs. The image then is put through contouring. Overlapping the two pictures, the vacant slots will be displayed green and the occupied ones red. Along with detecting if a slot is free or not, the code also detects incorrectly parked vehicles and can notify the user. With this application, the user can find a free parking spot in a single go instead of driving around and locating a spot manually; this saves a reasonable amount of fuel and decreases greenhouse gas emissions.**

## I. Introduction

There is an estimated number of 1.35 billion automobiles across the world. The increase in this number has been growing exponentially for the past ten years. The high production of automobiles has brought up issues regarding the comparatively fewer parking slots. Smart Parking is an intelligent parking system that assists drivers in finding a parking space. Using any smart device, the user will view a live feed of the parking lot. The application will assist the user in locating an empty spot. The camera captures a picture every time someone accesses the website. This helps solve significant problems like Fuel Wastage, which is the major leading cause of Green House Gas Emissions.

Also, the time wastage factor is reduced as now the driver knows exactly where he needs to go, and he does not need to go to 2-3 parking lots to find a space to park his vehicle. Furthermore, this increases safety as drivers are less distracted to look for parking spots and will not stop the cars in the middle of the roads. This will also reduce traffic on roads as people go towards a crowded parking lot, and the input-output buffer gets hugely crowded up, causing traffic jams. In the current parking system, there is no computerized system to store the parking details. All is done through manual entries. This causes problems such as time wastage and still does not guarantee parking for the driver.

On top of that, storing those manual entries on physical data means storage issues and data loss, and security concerns. Sometimes, vehicle users come to park their cars and find no space available, causing fuel wastage. Existing solutions include calling the parking centers to ask about availability, but this is unreliable as information some time ago becomes obsolete as parking spaces might get filled. However, an up-to-date real-time parking system solves the issue of wasting time, fuel, and one's patience. In the world of everything in the hands of technology, patience is no longer something people possess. It is time for even the parking system to upgrade to the needs of the people. Using OpenCV ( Open Source Computer Vision Library), we can automate the parking system completely. It uses stock empty parking images and relates to the current parking situation gives highly accurate results and real-time parking availability. It does not require a live camera that requires high power and continuous surveillance. Instead, it uses a traditional camera that updates with the website every time the user refresh. This all goes live, and anyone has to access the website to see the parking availability.

The intelligent OpenCV smart parking system is the right solution for highly-packed cities as users save time, fuel and prevents traffic jams. It provides the information required for parking, but the set-up is quite an issue as it has some prerequisites for the camera positioning, which may or maybe not be available in the parking lot.

## II. Literature Survey

Base Paper - Park Indicator: Book Your Parking Spot.

The paper is based on a simple implementation of the parking space booking system in Mumbai by creating a UI to collect and analyze data acquired from their users. The paper did not implement the project through image processing, however. Thus, with the inclusion of image processing concepts acquired, the project is automated and does not require users to update the database with parking vacancies across the different plots. References used include the OpenCV and python documentation to acquire knowledge regarding image processing functions, kaggle for datasets for testing the program, and a guide to work with latex files.

## III. Problem Statement

Our project idea is to develop a web application where users could book a parking spot, just like booking movie tickets to save people's time and fuel. This will relieve the congestion on urban city roads using image processing techniques such as edge detection and contouring.

## Objectives

With the help of this app, users can enter their required location, and based on that, and the app shows various parking lots nearby along with the vacancy status. The user then selects the desired parking lot, and the app displays exact parking spots with statuses of availability (red for occupied and green for vacant). Our system deals in real-time and will perform real-time monitoring of parking for both two and four-wheeled vehicles. The complete implementation of this project is explained in great detail in the further sections of this paper. This proposal aims to develop an application system through which the users could book their desired parking spot. The data is transferred in real-time between the application and the website operated by the parking lot admin. A standard database solution is proposed to collect the data, store it, and transfer it between the website and the application. All the data will be transmitted in real-time.

The main functions would include:

- Acquiring various parking lots' data
- Analyze and act upon the data
- Real-time parking availability status

## IV. Methodology

### A. System Architecture

Client-Server architecture is best suited for the project. A client-server application is a distributed system consisting of both client and server software. The architecture's significant components are App, Website, Central Database, Admin, Users.

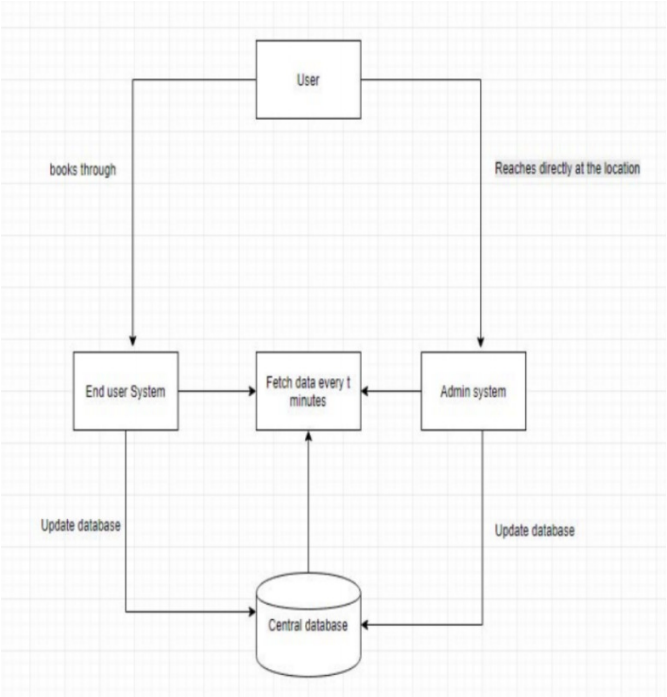


Fig. 1. System Architecture

### B. Working Mechanism

Algorithm for Python-based OPENCV image processing program -

1) Accept input images - empty plot and current live image
2) Resize the images to pre-set dimensions
3) Detect the edges of the empty plot image
4) Detect all possible edge combinations that can serve as parking spaces
5) Creating a differential image from the two given inputs
6) Detecting cars in the differential image
7) Checking if the cars overlap with the parking spaces found previously through edge detection
8) Return image with overlapped (non-empty) parking spaces in red and empty parking spaces in green

### C. Functions

The program is written in python language and uses the OpenCV module for its comfortable and straight forward image processing technique based functions for simplicity in writing and understanding the code.

*1) Edge Detection:* The function uses the Sobel edge detection technique based on the principle of convolution. It uses specific kernel matrices and implements convolution between the given image matrix and the kernel. 2 different Sobel edge detection instances are used to acquire horizontal and vertical lines present in the top view input image. The two are then combined to give an output that contains all possible edges: the plot lines present and some noise.

```
def edge (img):
    edge_x = sobel(img,1,0,kernel size = 3)
    edge_y = sobel(img,0,1,kernel size = 3)
    result = np.sqrt(edge_x**2 + edge_y**2)
    return result
```

*2) Contouring/Rectangle Identification:* The edge detection function's output serves as the input for this function. This function aims to recognize rectangles and, hence, parking spaces in the given input using the edges identified. The function uses inbuilt to find contours to identify all possible figures in the given image. The function also helps in the noise cancellation as it can identify only contours and rectangles of a required minimum area.

```
def contouring(img):
    contours = findContours (img)
    result = []
    for c in contours:
    if area(c) > minimum:
    add c to result
    return result
```

*3) Differential Image:* After identifying parking spaces in the given empty plot image, the live image of the plot is used to identify the parking spaces as vacant or occupied. This is done by only taking the absolute difference between the individual pixel values across the two input images' three color channels. The resulting differential image is would then be an image of just the vehicles present in the given plot. This new image is then passed through to the contouring function, which would identify the vehicles' locations with accuracy. This output would then be used to identify which parking spaces are then occupied and which are vacant.

```
def differential_image (empty, live):
    diff = absdiff (empty, live)
    diff, rectangle = contouring (diff)
    for r in rectangle :
        update empty with r as occupied
```

## V. Results and Analysis

The program can effectively analyze a given image and can give results with great success and hence the pros of the application include,

*1) Vehicle Identification:* It can identify any changes in the frame, i.e., objects present on the parking lot spaces, with great accuracy. It can identify any vehicles, objects, or even people present on a given parking space and can classify those spaces as occupied as they are unfit for parking purposes.

*2) Incorrect Parking:* When a vehicle is incorrectly parked by occupying two parking spaces instead of one, the program classifies both the parking spaces as occupied even though only one vehicle is present.


Fig. 2. Website View of Parking : Partially Filled


Fig. 3. Website View of Parking : Partially Empty

*3) Running Time:* The images inputted are resized to a minimum dimension so that they can be processed. This is done as most of the running times of the program's functions are influenced by the dimensions of the image, which, therefore, allows the program to run instantaneously.

*4) Simplicity:* Simplicity: The program and hence its application is relatively easy to understand. To exhibit


Fig. 4. Website View of Parking : Almost Empty


Fig. 5. Website View of Parking : Completely Packed

its practicality, if the program and application were to be launched in the public domain, even with many parking spaces being added to the domain, the program and application would be very user friendly due to its simple interface.

*5) Noise:* Unwanted signals do not affect the performance of the program.

However, several restrictions are in play in the operation of the program. The cons of the application include Image Clarity: Blurred or poor quality of images leads to low edge detection, resulting in a few parking spaces not being detected.

*Image Clarity:* Blurred or poor quality of images leads to low edge detection, resulting in a few parking spaces not being detected.

*Angle of Camera:* Image must be a top view for the program to classify plots accurately. This is not a hindrance in space identification, resulting in wrong outputs in the classification of spaces as occupied or vacant. This problem could be solved by automation of the program using ML/DL concepts but is not feasible using image processing techniques.

*Number of parking spaces:* In a given image, if the number of parking spaces reaches a maximum - which is around 300 currently, the program loses accuracy and fails to identify all plots and their vacancy statuses. This is primarily because the individual size of each parking space in the given image becomes very small and this makes it difficult for the program to not classify it as noise.

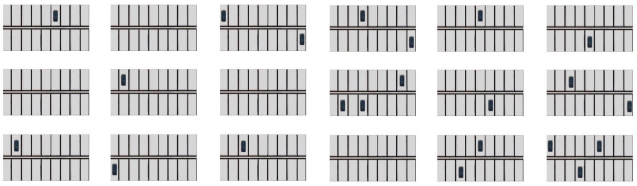

Fig. 6. Plot with 324 parking spaces

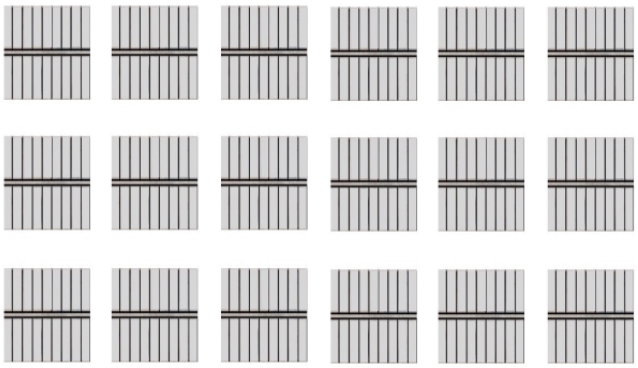

Fig. 7. Output with Fig 6 as Input

### A. Time Complexity analysis:

$$\text{Edge Detection} - O(m * n)$$

where m and n are the image dimensions.

$$\text{Differential Image} - O(m * n * 3)$$

where m and n are the image dimensions.

$$\text{Contouring} - O(m * n)$$

where m and n are the image dimensions.

### B. *Efficiency analysis:*

*Program Metric:* The metric used to evaluate the program is that of accuracy.

A database that follows the guidelines for the program has been used to calculate the accuracy of the program. The database uses several different plot images with plots of varying sizes and configurations and also uses several real life parking scenarios with differing degrees of parking availability.

The accuracy of the program calculated using our database is 94 per cent.

The program within its restrictions is very capable of achieving the required results. However, questions regarding efficiency can be raised when the images are outside its scope. The program is still able to function to some extent of success but is not perfect.

Feedback:

A - Excellent

B - Good

C - Satisfactory

D - Below Average

E - Poor

## VI. Conclusion

This project's main objective is to allow the user to view any parking lots' status, and based on the users' needs and requirements, and she/he could book a parking spot. Generally, the customers tend to go to the parking lots and then see if there is a vacancy, then they park their car, or else they find some other location. So instead, if the users already knew that the parking lot where the user is going to visit has a vacancy or not, and if not, then from the app, the user could find another parking spot at a different location. So our goal is to aid the users in finding and booking a parking spot, saving their time and fuel.

| Regarding | A | B | C | D | E |
|---|---|---|---|---|---|
| User Friendly | | ✓ | | | |
| Accuracy | | ✓ | | | |
| Image Flexibility | | | | ✓ | |
| Time for processing output | | ✓ | | | |
| Lack of bugs | | ✓ | | | |

Fig. 8. Efficiency Analysis

## Future Work:

1) ) Automation of processes by inculcating ML/DL concepts to reduce the restrictions on its functionality. This would also make the program more efficient than it currently is. If it is an application, then user feedback and its Five-point scale Likert analysis should be there (even if the feedback is exceptionally negative, you can mention it along with its reason).
2) Adding parking costs for individual plots.
3) Adding directions to an empty parking space with the help of the Google Maps API.

## VII. Acknowledgment

The authors would like to acknowledge our guide, Mr. Ramu's support, and we would also like to acknowledge numerous discussions on the subject with him.

## VIII. Individual contribution

### A. *Soorya*

- Research
- Report Writing
- Dataset Pre-Processing
- Testing
- Differential Image function

### B. *Ayush*

- Report Writing
- Latex documentation
- Dataset Pre-Processing
- Web-App development
- Presentation

### C. *Ibrahim*

- Research
- Report Writing
- Edge detection
- Contour function
- Web-App development

- Presentation

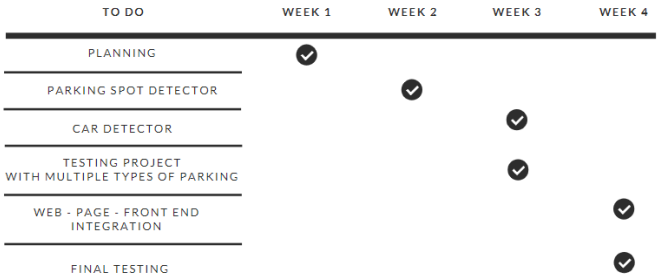


Fig. 9. Gaant Chart of Progress

## IMPLEMENTED/BASE PAPER

Park Indicator: Book Your Parking Spot by Sagar Piyush Parikh, Dhrumil Rakesh Shah and Parshva Rajesh Shah during Fourth International Conference on Computing Communication Control and Automation in 2018

## REFERENCES

[1] OpenCV Documentation

[2] kaggle.com - for datasets

[3] H. Kopka and P. W. Daly, A Guide to LATEX