# PARKING SPACE DETECTION

**A PROJECT REPORT**

*for*

**Artificial Intelligence**

*in*

**B.Tech (IT)**

*by*


**Mugilan A (19BIT0102)**

**Mehar Girdhar (19BIT0113)**


**Fall Semester,2021-22**

*Under the Guidance of*

**Dr. R.Subhashini**

SITE

**School of Information Technology and Engineering**

DEC, 2021

## DECLARATION BY THE CANDIDATE

We here by declare that the project report entitled **"PARKING SPACE DETECTION"** submitted by us to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Artificial Intelligence (ITE2010)** is a record of bonafide project work carried out by us under the guidance of DR**. R SUBHASHINI.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place: Vellore                                                                 Signature

Date: December 10, 2021

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Information Technology & Engineering [SITE]**

## CERTIFICATE

This is to certify that the project report entitled **“PARKING SPACE DETECTION”** submitted by **Mugilan A (19BIT0102), Mehar Girdhar (19BIT0113)** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Artificial Intelligence (ITE2010)** is a record of bonafide work carried out by them under my guidance.

**DR.R SUBHASHINI** **GUIDE** **SITE**

# PARKING SPACE DETECTION

**Mugilan A[1], Mehar Girdhar[2]**

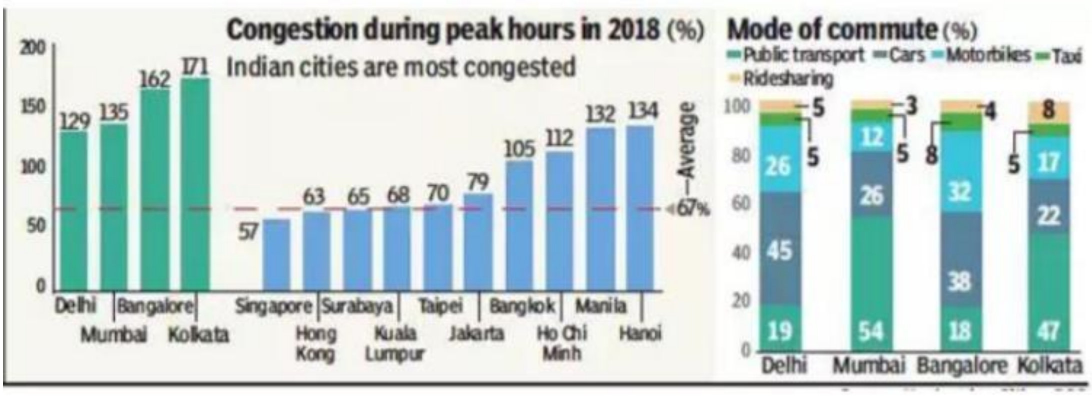
[1,2] **Department of Information Technology, VIT University, Vellore, Tamil Nadu, India**

**Abstract**

*Drivers often encounter problems associated with locating empty parking slots in parking areas. This paper presents a smart parking lot management system which operates using image processing. An image processing algorithm is used to detect empty parking areas from aerial images of the parking space. The algorithm processes the image, extracts occupancy information concerning spots, and their positions thereof. The system also reports if individual parking spots are occupied or otherwise. Occupancy information is made available to newly arriving drivers by projecting it unto large displays positioned at vantage points near the vicinity. The smart parking lot management system reduces the stress and time wastage associated with car parking and makes management of such areas less costly. Our project targets small scale Business which has fewer parking slots to provide a cost efficient and simple method for detecting parking slot.*

## I. INTRODUCTION

Shopping centers and Malls are facing a decline in sales in India due to online shopping apps like Flipkart, Amazon etc. and as a result many consumers find it more convenient to buy products online, rather than face hefty lines of traffic in and out of shopping malls and especially in parking lots. Once in the lot, they must cruise up and down multiple floors of full spaces searching for an available spot, which may be far away from their actual intended shopping destination. This is the reason why consumers avoid brick and mortar shopping as much as they can. Below figure shows parking congestions as on 2018 in metro cities




*Source: Unclogging Cities, BCG*

With the changing behavior of people, shopping malls operators are coming under increased pressure to find solutions to such car parking problems in their large shopping malls to improve and enhance the traditional shopping experience and drive revenue. Many commercial operators are turning to new technologies like artificial intelligence that leverage data to analyze traffic, identify drivers of customer satisfaction and increase profits. To solve this, we have applied smart parking detection technology to gather and analyze parking statistics, hence retail operators have a powerful tool to optimize the parking space and increase customer experience and maximize parking and retail revenue – turning their properties into optimal revenue generating centers. That tool is detection of proper parking of car in parking spaces by just using preinstalled security cameras.

## II. BACKGROUND

### *New Learning experience gained*

**Image-Processing**

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then "Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality".

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of colored images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

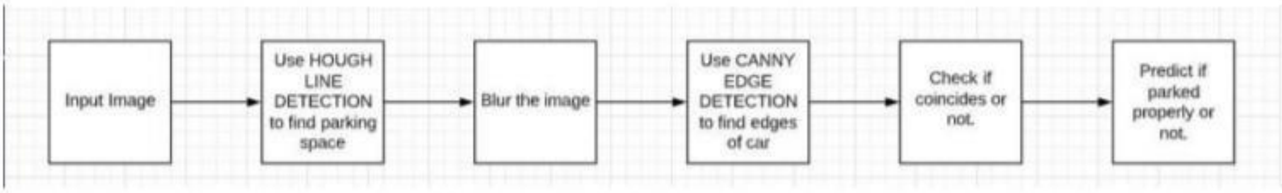
1. Importing the image
2. Analyzing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis

**OpenCV:**

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

We also learnt implementation of Algorithms like Hough Transformation and Canny edge detection in OpenCV

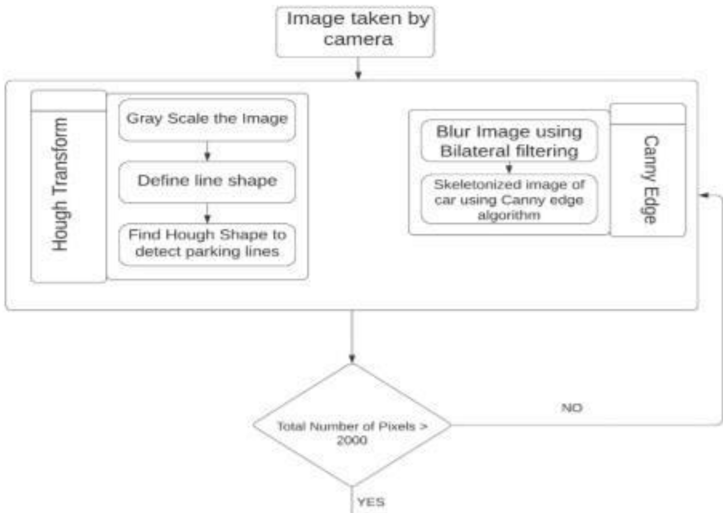
## *Architecture*




When the car is parked, image is captured by the camera in the parking lot. A picture is capture from the video feed when parking detection is required, otherwise if we keep capturing images until they are required, it will increase computation cost and we cannot afford that. Now we use OPENCV on the image. After this, we greyscale the image as it is a pre-requistite of detecting shapes using Hough transform. Now we define parametric line as we have to detect them. After this, using the Hough lines detection, the painted parking division lines will be detected (in red color) and their x-coordinates will be noted (x1, x4). Now, to use Canny Edge detection, we will first blur the image using Bilateral filtering. This will assist to skeletonize the car. The x-coordinate extremities (x2, x3) of the car will be noted. Using python inequality, we will check whether (x2, x3) lies between (x1, x4). Then, we also need to check whether the number of non-zero pixels are greater than 2000 or not (hit and trial). If so happens, the car is properly parked otherwise the car is not properly parked.

- The security camera captures the image of the car parked and then we kick in some OPENCV.
- Using Hough line detection algorithm, we will detect the parking space where the cars to be parked.
- We will blur the image so that the edge detection can be carried out.
- Then, using Canny edge detection algorithm we will draw out the edges of the car.
- Finally, we check whether the extremities of the x-axis of edges of car lies between the detected Hough lines or not.
- If parked improperly, email the user.

*Architecture Diagram*




## III. Literature Survey:

Road detection is very important when we talk about the field of remote sensing image analytics **[1]**. The normal Hough transform can easily do the job when we talk about straight road detection. One such algorithm is generated where such transform can be considered as a linear transform which accelerates the parallel implementation of this transform on multiple images. With such algorithm, this transform can even have better accuracy than the traditional Radon transform. This is used in UAV’s software’s which can help spy enemy territories by giving away the location of roads with very high computation speed per image. The traditional UAVs do not have such software and must click images which are then analyzed back in the base. With such algorithm, it can segregate among multiple images in air itself.

Sometimes **[2]** images often get subjected to speckle noise and other degradations which are high frequency components. For segmentation, edge detection is used to get more precise measurements of elements in the picture. At times, filters degrade edges when used for noise removal which causes problems in edge detection algorithms. Canny operator is a famous edge detector in many fields but it causes softening of edges. To prevent this, a modified algorithm where a modified median filter is placed instead of Gaussian smoothing element in the Canny edge detection algorithm which removes such high frequency components like speckle noise with little degradation of edges. A weak weighted smoothing filter is also applied which in a controlled way removes all other noise with a negligible sacrifice to the edges. Such improved algorithm is used in factories and even in the medical industry.

The default Hough transform sometimes fails to meet the real-time requirements and this can pose a problem in the field of vehicle safety and navigation as we are taking a leap from normal cars to AI controlled cars **[3]**. To accelerate the accuracy of Hough Transform, various attempts are being made. In one such method, the additive property of Hough transform is used in multiple levels of the hierarchical Hough transform. The complex operations are replaced with simpler additions, and this in return, reduces the number of points processed at the lowest level of hierarchy. This new method is called hierarchical additive Hough transform (HAHT). This leads to drastic reduction in the computational complexity of the voting step (one step out of two) as compared to the existing hierarchical approach of the Hough transform. If we assume that the thresholding scheme of Hough transform is employed for the HAHT, then the peak detection processes would be similar in terms of complexity. On improving the algorithm, it can even detect curved lanes by assuming the curve to just be a set of straight lines and also keeping the angle measurements in the mind of these line segments which can be a breakthrough on the field of artificial intelligence.

A **[4]** smart parking system can be created using image processing and artificial intelligence. Cameras and sensors can be deployed in the parking lot to recognize the license plate numbers, ensuring ticketless process. To this, neural networks and big data analysis can be included in the algorithm to have parking information and recommendations. Raspberry Pi is the hardware required where the sim available will communicate with the cloud. The cloud will contain entries of prior parking and new algorithms can be used for recommendations. Using multiple technologies, a fully automated parking garage can be made with hassle free experience

With the improvement of innovation **[6]**, shrewd gadgets are getting more normal in regular day to day existence. The advancement of gadgets that can interface with the Internet and communicate information has been a wellspring of motivation for brilliant city plans. The regular issue in the urban communities is the trouble of discovering free stopping openings. The stopping issue makes traffic block and individuals who go to work are searching for a spot. In this investigation, a route and reservation-based stopping proposition framework was created for brilliant urban areas. The proposed strategy includes the improvement of little gadgets that send information to the web utilizing the web of things (IoT) innovation. The free parking spot nearest to the current area is found by hereditary calculation. The proposed technique is tried for various situations and exact outcomes are acquired.

The customary Canny edge **[7]** discovery technique is generally utilized in dim picture preparing. Be that as it may, this customary calculation can't manage shading pictures and the boundaries in the calculation are hard to be resolved adaptively. In this paper, an improved Canny calculation is proposed to recognize edges in shading picture. The proposed calculation is made out of the accompanying advances: quaternion weighted normal channel, vector Sobel inclination calculation, non-maxima concealment dependent on introduction, edge recognition and association. Trial results show that the proposed calculation outflanks other shading picture edge discovery strategies and can be broadly utilized in shading picture preparing.

Hough Transformation (HT) **[8]** is a productive technique to distinguish straight lines in advanced pictures. In the regular HT, pixel contiguity isn't considered, and this prompts the accompanying disadvantages: (1) genuine length of line sections can't be registered; (2) colinear line portions can't be recognized; and (3) frequently, bogus lines are distinguished and short lines go undetected. This paper proposes an adjusted Hough Transformation which performs contiguity check in a straightforward and effective manner. A systolic engineering that executes this changed change is introduced. The systolic cluster takes the digit guide of the parallel picture as information and cycles one line/segment of pixels simultaneously. The territory time intricacy of the proposed engineering is demonstrated to be better than the traditional consecutive calculation. Primer reproduction results are introduced.

The article **[9]** presents an intelligent system for parking space detection based on image processing technique. The proposed system captures and processes the rounded image drawn at parking lot and produces the information of the empty car parking spaces. In this work, a camera is used as a sensor to take photos to show the occupancy of car parks. The reason why a camera is used is because with an image it can detect the presence of many cars at once. Also, the camera can be easily moved to detect different car parking lots. By having this image, the particular car parks vacant can be known and then the processed information was used to guide a driver to an available car park rather than wasting time to find one. The proposed system has been developed in both software and hardware platform. An automatic parking system is used to make the whole process of parking cars more efficient and less complex for both drivers and administrators.

The article **[10]** presents a comprehensive analysis on crucial aspects for designing a smart parking system such as sensor selection and optimal position for sensor deployment for accurate detection. Initially, two most common sensor, Light Dependable Resistor (LDR) sensor that works on shadow detection principal and Infra-Red (IR) sensor which works on object detection mechanism are used. The performance analysis of the accuracy for detection of vacant parking slots and vehicle detection under different conditions is presented. It is concluded that IR sensor outperforms LDR sensor in terms of it's accuracy in detecting the vacant parking slots and vehicle detection in different environmental factors.

The article **[11]** describes a novel vehicle detection sensor design in which dual microwave Doppler radar transceiver modules were used to detect the movement of a parking vehicle. A motion recognition algorithm was also presented to identify the vehicle behavior and generate the parking space occupancy status. Comparing with existing methods such as magnetometer and optical based detection, the proposed design simplified engineering integration from complex optical system design as well as achieved a high detection accuracy. Experimental results showed that the proposed dual microwave Doppler radar sensor detected the vehicle movement clearly and the parking space occupancy detection accuracy was higher than 98%.

The article **[12]** deals with parking space detection by using ultrasonic sensor. Using the multiple echo function, the accuracy of edge detection was increased. After inspecting effect on the multiple echo function in indoor experiment, we applied to 11 types of vehicles in real parking environment and made experiments on edge detection with various values of resolution. We can scan parking space more accurately in real parking environment. We propose the diagonal sensor to get information about the side of parking space. This method has benefit calculation and implementation is very simple.

This article**[13]** analyzes the current situation of intelligence transport and proposes a design of video-based parking space detection system, gives a program from the hardware and software platform structure to the detection algorithm design process. And finally gives the results of field tests at the scene. The design may provide some references to the current and the future intelligence transport research, or particularly, to the management of smart parking lots.

| Name of Algorithm | Advantages of the technique | Disadvantages of the Technique |
|---|---|---|
| Radon Transform [1] | Low computational complexity, Less time consumption | Smearing, data loss, Creating a non-unique solution from unique input data |
| Improvised canny Edge detection [2] | Can preserve more useful edge information, more robust to noise | Computationally intensive, Time consuming |
| Hierarchical additive Hough Transformation [3] | Computationally less intensive, Savings in time costs | Can give misleading results when objects happen to be aligned by chance |
| Platforms such as Node-RED, OpenALPR [4] | Flexible, High accuracy than simple methods of detection | Multiple technologies are required to implement the method |
| Hough Transform [9] | Can detect moving images, Fast and efficient Less computationally intensive | Susceptible to weather changes |

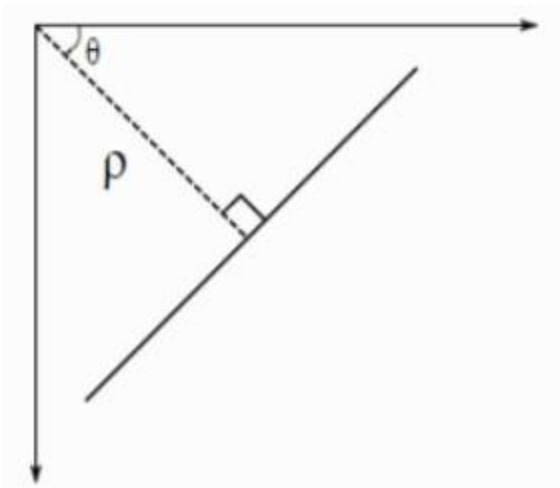
| Parking space detection techniques | Advantages of the technique | Disadvantage of the technique |
|---|---|---|
| Infrared Detection [10] | Conditions in the fog have a longer detection range than visible light, Less impact by day-night conversion | Affected by the rain and snow and airflow change |
| microwave detection [11] | Keep stability under adverse weather conditions, can detect a number of parking spaces Small transmitting power | High cost |
| ultrasonic detection [4,12] | Small size, easy to install, long life | Can be affected by air, Temperature changes affect the performances, The non-vehicle object in the detection region is likely to interfere with |
| video image detection [9,13] | Image information content is | The accuracy of detection is attached to the location of |

| | | |
|---|---|---|
| | very rich,<br>Easy to install,<br>Adjust and move the location of detector. Low cost | camera placement |

## IV. PROPOSED ALGORITHM

### *1. Hough transform*

It is used to find shapes (here, lines) in an image. Suppose, a line represented as $\rho = x.\cos\theta + y.\sin\theta$ (or as $y = mx + c$). Here ρ is the perpendicular distance from origin to line, and θ is the angle formed by this perpendicular and horizontal axis measured in counter-clockwise (used by openCV).




In image space, a line is represented as x vs y. But this line in Hough space is represented as a dot where x-axis is m and y-axis are b. If the line to be represented has parameters, then a graph is plotted where x-axis is ρ and y-axis is θ. A voting system is followed as many times values get repeated and a place with higher count is considered to be having a line in it. Firstly, we make a 2 X 2 matrix (called accumulator) with ρ as rows and θ as columns.
Initially, it is zero matrix. Now, we select a point on the line and then we keep changing the θ from 0, 1, 2, …, 180 and then we check the value of ρ again. If kept in simple words, we have the value of (x, y) in $\rho = x.\cos\theta + y.\sin\theta$ and we keep changing the value of θ thereby obtaining the value of ρ each time. Now, for every (ρ, θ), increment the value of accumulator by 1 and then the same process is followed. The line will be detected when it will always have a specific value of (ρ, θ) or in simple words, if an accumulator in Hough space has very high votes. All this is available in the OpenCV function, cv2.HoughLines(). It returns an array of (ρ, θ) values. Here, ρ is in pixels and θ is in radians.

### *2. Blurring the image (SMOOTHING)*

Image blurring is executed by using a low-pass filter kernel on it and in return it removes noise by removing high frequency content from the image. In this process, edges also get blurred a little bit. In this project we have used Bilateral Filter as it is proved to be highly effective in noise removal and it also keeps the edges in the image sharp with a little sacrifice of runtime (when compared to other noise removing filters).

### *3. Canny Edge Detection Algorithm*

This algorithm is a popular edge detection algorithm which if kept in simple words, creates an image with dominant edges. It acts like an upgrade to Sobel operator and keeps valuable edges and removes others (uses 2-step thresholding to do so). It thins up all the edges until they are like 1 pixel wide as thick edges are of no use to us. The process is explained in detail below:

#### *3.1 Intensity Gradient of the Image*

In the first step, Sobel kernel is used in x (horizontal direction) and y (vertical) direction to obtain Gx (derivative of x) and Gy (derivative of y) respectively. This will completely filter out the smoothened pixels. After this, we find edge gradient and direction for each pixel using the formula:

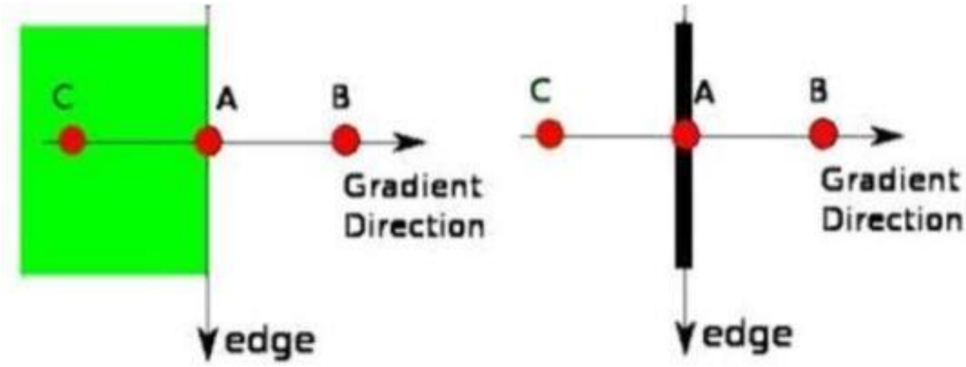
$$G = \sqrt{G_x^2 + G_y^2}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

It is to be noted that the gradient direction is perpendicular to the edges. This estimation is correct to one of four angles each representing vertical, horizontal and also the two diagonal directions.

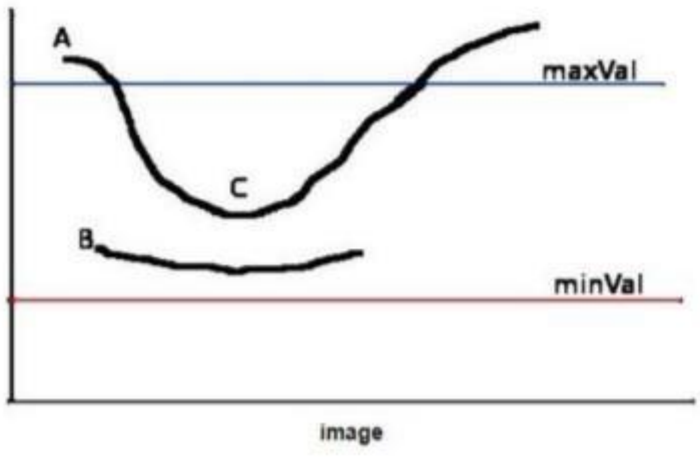
#### *3.2 Non-maximum Suppression*

Now that we have the value of gradient magnitude as well as direction, the image is now made to scan such that unwanted pixels can be eliminated (pixels that do not constitute in making an edge). To execute the following, each and every pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. This will assist in getting thinner edges. For this, kindly refer the image below:




### *3.2 Hysteresis Thresholding*

This is the final step which rectifies all the edges into actual edges and makes sure they are not some other artifact. For this, two threshold values are required namely, minVal and maxVal. Edges having intensity gradient more than maxVal have a greater chance to be edges and those edges below minVal are non-edges and are discarded. Also, the edges which lie between these two thresholds values are considered edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are counted as edges otherwise they are also discarded. This is explained using the image below:



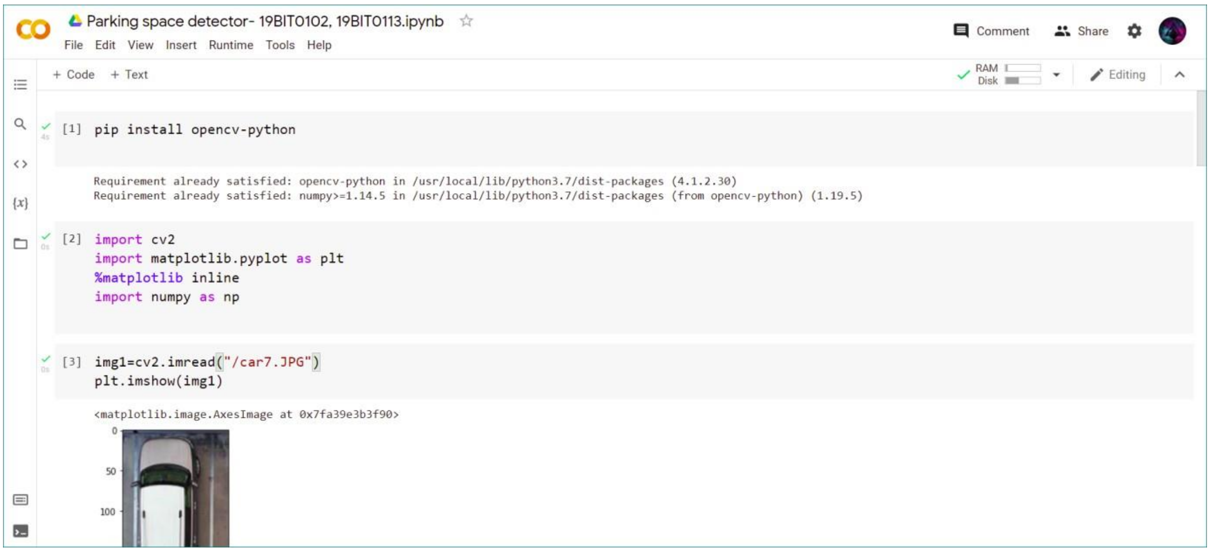

In the given diagram, A here is the maxVal and is hence a sure-edge. Now, edge C is below maxVal but since it is connected to edge A, it is also considered as valid edge and hence we get full curve. For edge B, although it is above minVal value and lies in same region as that of C, it is not joined to any "sure-edge" and is hence discarded. Hence, the selection of values of minVal and maxVal is really important to get the correct result. A value too low will select all edges and a very high value will discard edges lying between maxVal and minVal. In this stage, small pixel noises are also removed on the assumption that edges are long lines. So, in the end, what we finally get are strong edges present in the image.

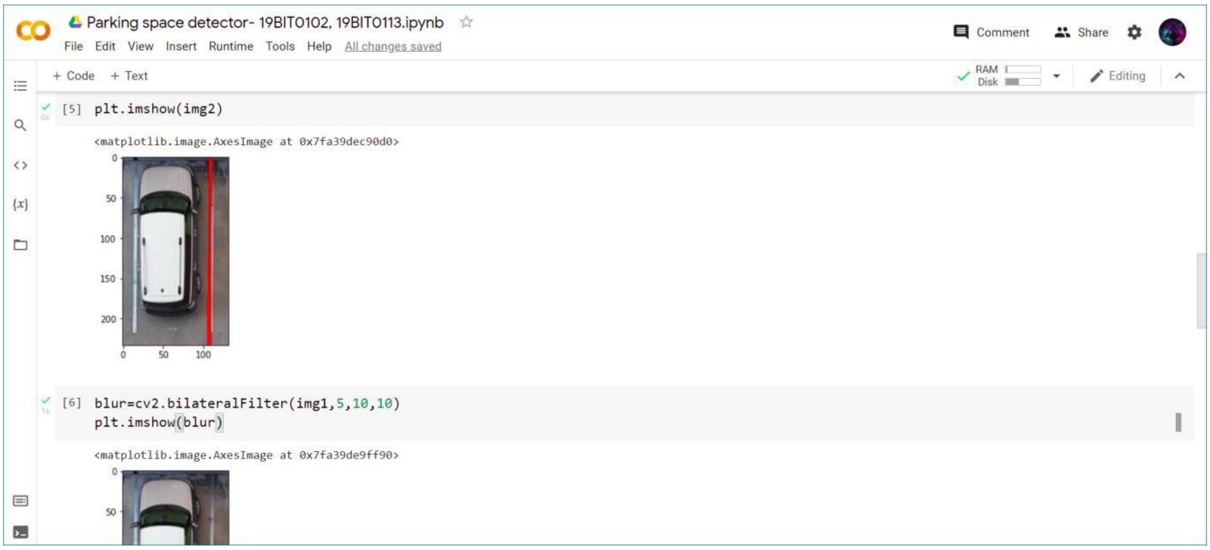
***Hardware and Software requirements***:

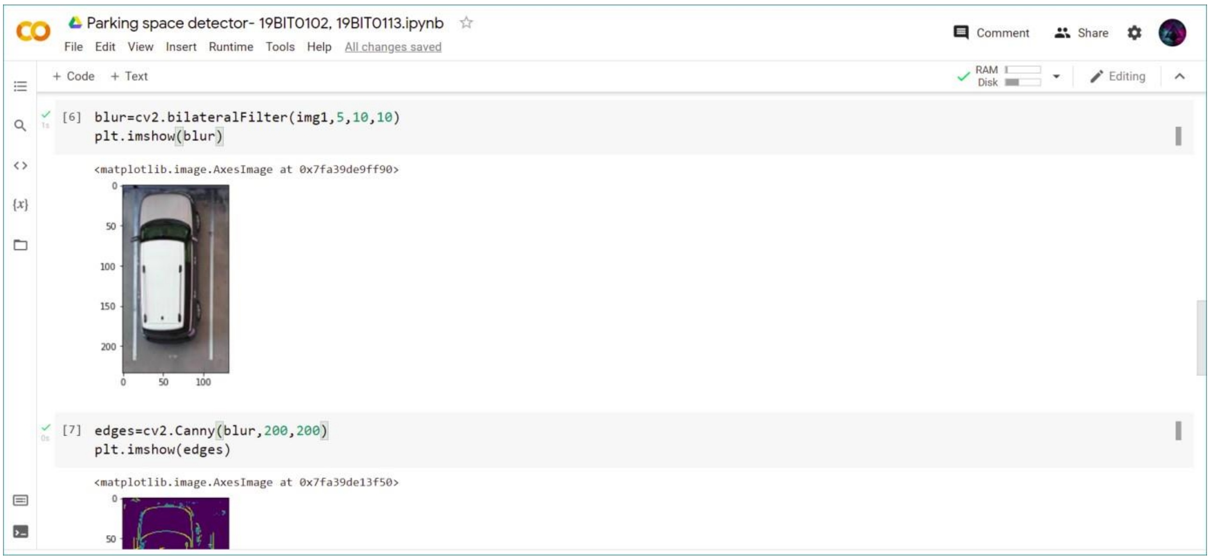
- Computer Vision API: OPEN_CV
- CV Functions: BLUR, CANNY, HOUGH_LINES
- CAMERA IN PARKING LOT (TOP VIEW)

# EXPERIMENTS RESULTS

**Parking occupied:**





Parking space detector- 19BIT0102, 19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk Editing
[5] plt.imshow(img2)
<matplotlib.image.AxesImage at 0x7fa39dec90d0>
[6] blur=cv2.bilateralFilter(img1,5,10,10)
plt.imshow(blur)
<matplotlib.image.AxesImage at 0x7fa39de9ff90>


Parking space detector- 19BIT0102, 19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk Editing
[6] blur=cv2.bilateralFilter(img1,5,10,10)
plt.imshow(blur)
<matplotlib.image.AxesImage at 0x7fa39de9ff90>
[7] edges=cv2.Canny(blur,200,200)
plt.imshow(edges)
<matplotlib.image.AxesImage at 0x7fa39de13f50>

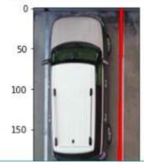

Parking space detector- 19BIT0102, 19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk Editing
[7] edges=cv2.Canny(blur,200,200)
plt.imshow(edges)
<matplotlib.image.AxesImage at 0x7fa39de13f50>
[8] arr=[]
for i in range(edges.shape[1]):
for j in range(edges.shape[0]):
if edges[j][i]!=0:
arr.append(i)

Parking space detector- 19BIT0102, 19BIT0113.ipynb

```
[8] arr=[]
    for i in range(edges.shape[1]):
      for j in range(edges.shape[0]):
        if edges[j][i]!=0:
          arr.append(i)
```

```
[9] if len(arr)>2000:
      print("parking occupied")
    else:
      print("parking available")
    plt.imshow(img2)
```

```
parking occupied
<matplotlib.image.AxesImage at 0x7fa39dd7bc90>
```




## Parking Unoccupied:

Parking space detector-19BIT0102,19BIT0113.ipynb

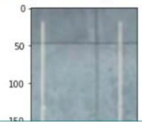
```
[21] pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python) (1.19.5)
```

```
[22] import cv2
     import matplotlib.pyplot as plt
     %matplotlib inline
     import numpy as np
```

```
[23] img1=cv2.imread("/nocar1.JPG")
     plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7fbc11aec7d0>
```

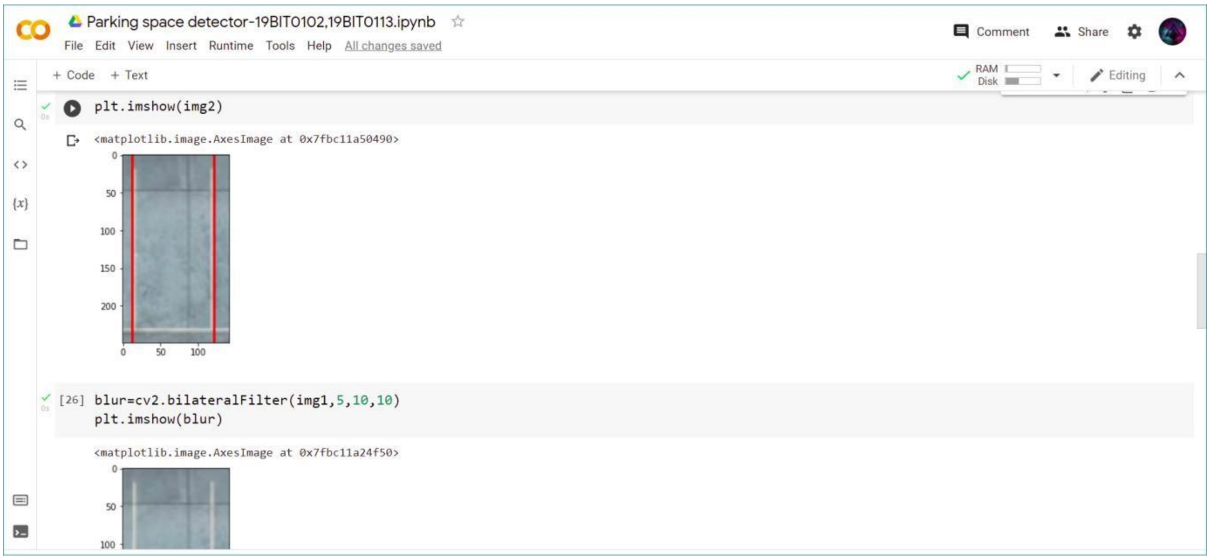


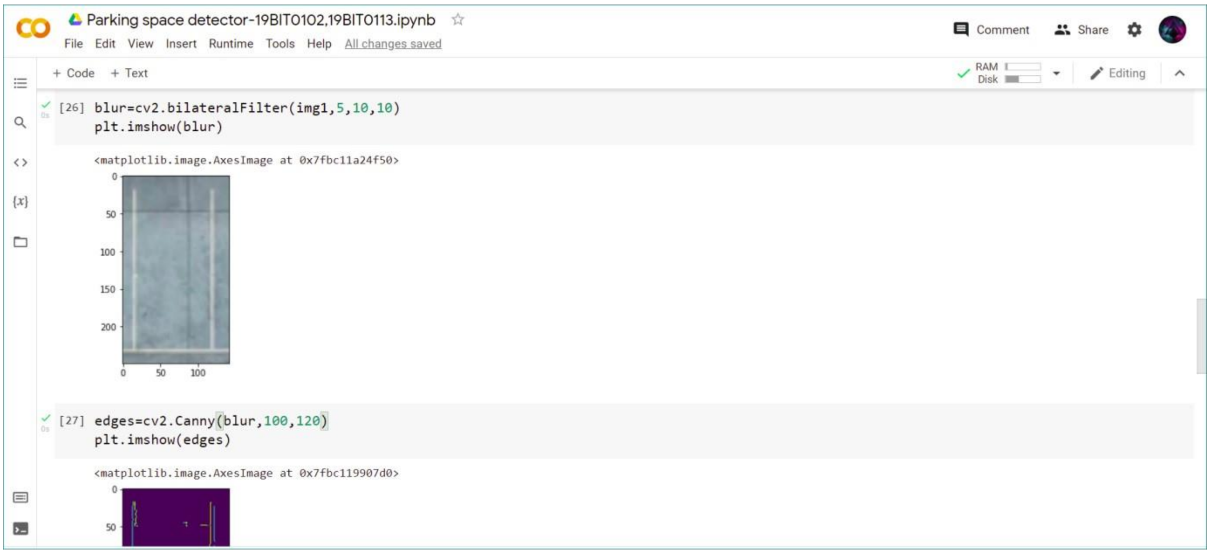

Parking space detector-19BIT0102,19BIT0113.ipynb

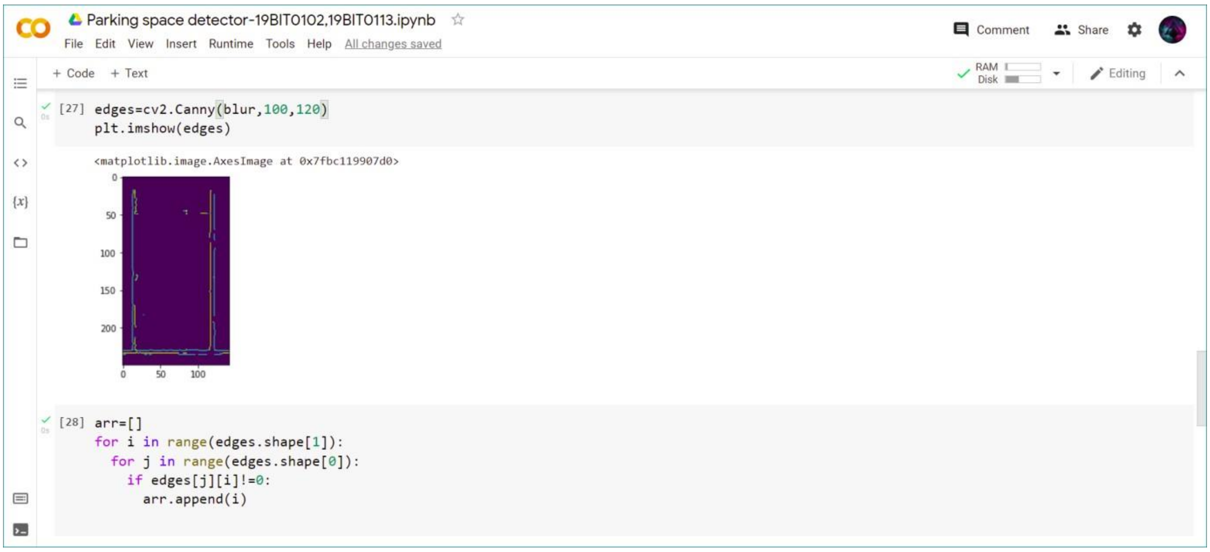
```
[24] img2=cv2.imread("/nocar1.JPG")
     gray=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
     pts=[]
     for apperturesize in [3,7]:
       edges=cv2.Canny(gray,50,100,apertureSize=apperturesize)
       lines=cv2.HoughLines(edges,1,np.pi/180,100)
       for rho,theta in lines[0]:
         a=np.cos(theta)
         b=np.sin(theta)
         x0=a*rho
         y0=b*rho
         x1=int(x0+1000*(-b))
         y1=int(y0+1000*(a))
         x2=int(x0-1000*(-b))
         y2=int(y0-1000*(a))
         cv2.line(img2,(x1,y1),(x2,y2),(255,0,0),2)
         pts.append(x1)

     print(pts)
```

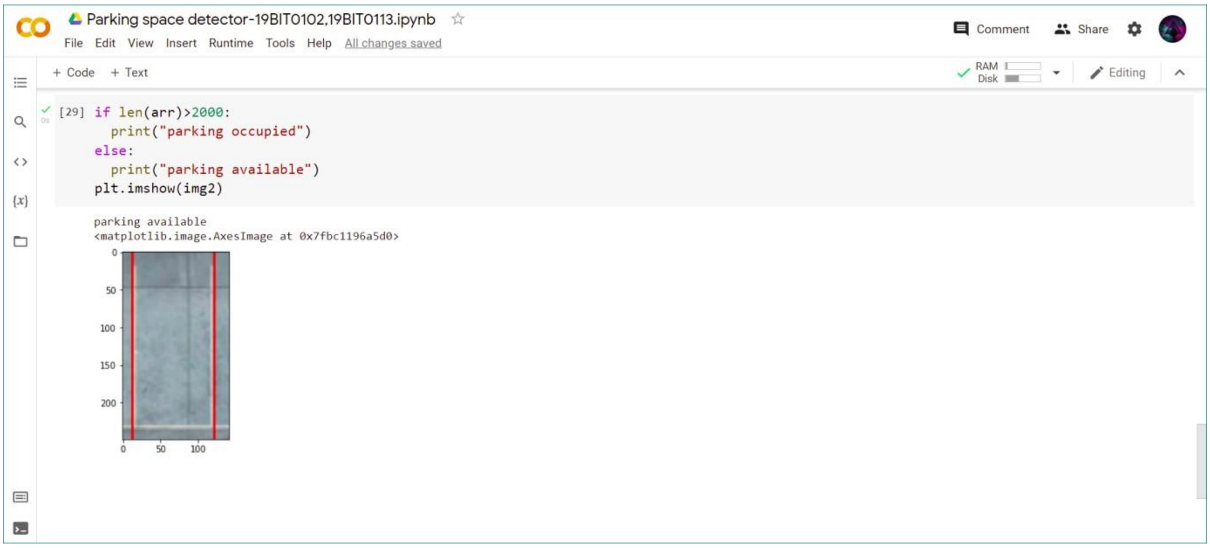
```
[121, 13]
```

```
[25] plt.imshow(img2)
```


Parking space detector-19BIT0102,19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
plt.imshow(img2)
<matplotlib.image.AxesImage at 0x7fbc11a50490>
[26] blur=cv2.bilateralFilter(img1,5,10,10)
plt.imshow(blur)
<matplotlib.image.AxesImage at 0x7fbc11a24f50>


Parking space detector-19BIT0102,19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[26] blur=cv2.bilateralFilter(img1,5,10,10)
plt.imshow(blur)
<matplotlib.image.AxesImage at 0x7fbc11a24f50>
[27] edges=cv2.Canny(blur,100,120)
plt.imshow(edges)
<matplotlib.image.AxesImage at 0x7fbc119907d0>


Parking space detector-19BIT0102,19BIT0113.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[27] edges=cv2.Canny(blur,100,120)
plt.imshow(edges)
<matplotlib.image.AxesImage at 0x7fbc119907d0>
[28] arr=[]
for i in range(edges.shape[1]):
for j in range(edges.shape[0]):
if edges[j][i]!=0:
arr.append(i)




## Video Presentation Link:

https://drive.google.com/file/d/1gSqd09u6P1er5GTP8p0usuUHbyz4oWMI/view?usp=sharing

## COMPARATIVE STUDY

Most of the solutions present in the industry use large dataset to identify the type of the car and read the number plate but using that is very costly in terms of computational resources required. Also, due to the presence of large dataset, the model becomes slow. In our project, We are using the basic Bounding Box around a car to know where a car is so that we can find the incorrectly parked cars in a parking lot. This solution uses very low computational resources and will act as a game changer and is very helpful for small shopping centers existing

## CONCLUSION AND FUTURE WORK

With increasing number of cars in India the problem of parking is also increasing. We are trying to use the camera installed in parking lot to detect free parking spaces. This project targets small offices and shops which cannot afford manual labor and has small sized parking space. The solution we are working on uses Computer Vision to identify the parking spaces which are occupied and those that are free so as to effectively manage parking space.
For this we will be taking advantage of OpenCV library of Python programming language.
The Future works of our project includes Detecting parking slots in moving videos and identifying parking spaces in different angles.

## REFERENCES

[1] Weifeng Liu, Zhenqing Zhang, Shuying Li, “Dapeng Tao, Road Detection by Using a Generalized Hough Transform”, Remote Sens. 2017, 9, 590; doi:10.3390/rs9060590,www.mdpi.com/journal/remotesensing

[2] Weibin Rong, Zhanjing Li, Wei Zhang and Lining Sun, “An Improved Canny Edge Detection Algorithm”, Proceedings of 2016 IEEE International Conference on Mechatronics and Automation August 3 - 6, Tianjin, China

[3] Ravi Kumar Satzoda, Suchitra Sathyanarayana, ThambipillaiSrikan than, SupriyaSathyanarayana, “Hierarchical Additive Hough Transform for Lane Detection”, IEEE EMBEDDED SYSTEMS LETTERS, VOL. 2, NO. 2, JUNE 2015

[4] Jiang Ruili, Wang Haocong, Wang Han, Dr Eoin O’Connell, Dr Sean McGrath, “Smart Parking System Using Image Processing and Artificial Intelligence”, 978-1-5386-5147- 6/18/$31.00 ©2018 IEEE

[5] Mei, Z., Zhang, W., Zhang, L., & Wang, D. (2020). Optimization of reservation parking space configurations in city centers through an agent-based simulation. Simulation Modelling Practice And Theory, 99, 102020. https://doi.org/10.1016/j.simpat.2019.102020

[6] Aydin, I., Karakose, M., & Karakose, E. (2017). A navigation and reservation based smart parking platform using genetic optimization for smart cities. 2017 5Th International Istanbul Smart Grid And Cities Congress And Fair (ICSG). https://doi.org/10.1109/sgcf.2017.7947615

[7] Xin, G., Ke, C., & Xiaoguang, H. (2012). An improved Canny edge detection algorithm for color image. IEEE 10Th International Conference On Industrial Informatics. https://doi.org/10.1109/indin.2012.6301061

[8] Li, H., Pao, D., & Jayakumar, R. (1989). Improvements and systolic implementation of the hough transformation for straight line detection. Pattern Recognition, 22(6), 697-706. https://doi.org/10.1016/0031-3203(89)90006-x

[9] Al-Kharusi, H., & Al-Bahadly, I. (2021). Intelligent Parking Management System Based on Image Processing. Retrieved 29 November 2021, from http://www.scirp.org/journal/PaperInformation.aspx?PaperID=45644

[10] Bachani, M., Qureshi, U., & Shaikh, F. (2016). Performance Analysis of Proximity and Light Sensors for Smart Parking. Procedia Computer Science, 83, 385-392. https://doi.org/10.1016/j.procs.2016.04.200

[11] Bao, X., Zhan, Y., Xu, C., Hu, K., Zheng, C., & Wang, Y. (2017). A novel dual microwave Doppler radar based vehicle detection sensor for parking lot occupancy detection. IEICE Electronics Express, 14(1), 20161087-20161087. https://doi.org/10.1587/elex.13.20161087

[12] Wan-Joo Park, Byung-Sung Kim, Dong-Eun Seo, Dong-Suk Kim, & Kwae-Hi Lee. (2008). Parking space detection using ultrasonic sensor in parking assistance system. 2008 IEEE Intelligent Vehicles Symposium. https://doi.org/10.1109/ivs.2008.4621296

[13] Bin, Z., Dalin, J., Fang, W., & Tingting, W. (2009). A design of parking space detector based on video image. 2009 9Th International Conference On Electronic Measurement & Instruments. https://doi.org/10.1109/icemi.2009.5274579

[14] Find where to park in real time using OpenCV and Tensorflow. Medium. (2021). Retrieved 25 November 2021, from https://towardsdatascience.com/find-where-to-park-in-real-time-usingopencv-and-tensorflow-4307a4c3da03.