International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Smart Parking System Using Image Processing

Shruthi B[1], Raghottam J Kulkarni[2], Sanath Kumar S[3], Dilip Kumar S[4]

[1]*Assistant Professor, Dept. of ISE, Atria Institute of Technology, Bengaluru, Karnataka, India*
[2,3,4]*Undergraduate Student, Dept. of ISE, Atria Institute of Technology, Bengaluru, Karnataka, India*

Abstract - *This paper presents an image-processing based smart parking system developed for multi-storey parking garages, open parking lots and many more. The proposed system design through the Python and the OpenCV library utilizes the combined edge detection and coordinate bound pixel sections in deciding if a parking spot in the acquired footage is occupied or not. It also presents the implementation of image to text conversion. Tesseract is used for text extraction from the processed image. The variable level of image processing ensures that different images get different levels of treatment in order to produce optimized text results.*

Key Words: **OpenCV, Tesseract, Image Processing, Optical Character Recognition, Multi-Storey Structures, Cloud Firestore**

1. INTRODUCTION

Globalization has continually accumulated more individuals into urban territories causing significant urban communities like Bangalore to be vigorously populated and blocked. The expansion in population implies an expansion additionally in human mobility. This influences the expansion in the number of vehicles which thus influences the parking circumstance. These days, a few people are purchasing vehicles regardless of whether they have no place to put them, and a few streets are in any event, turning out to be parking spaces which at that point causes substantial traffic. Ordinary parking regions are typically simply vacant spaces, and individuals needed to search for an empty one manually. Not just that this parking technique is very tedious, it is also inefficient particularly for multi-storey structures where drivers need to survey each spot and experience different floors just to find and make sure about a parking spot.

1.1 OpenCV

OpenCV (Open Source Computer vision) is permitted for both scholastic and commercial use. It is a library of programming functions mainly aimed at real-time computer vision. OpenCV's application has wide areas which includes 2D and 3D feature toolkits, Ego motion estimation, Facial recognition system, Gesture recognition, Motion understanding, Object identification Segmentation and recognition and Motion tracking. OpenCV is written in C++ and its primary interface is in C++, however it despite retains a less comprehensive though extensive older C interface. OpenCV contains libraries of pre-defined functions supportive in image processing. Since it is open source, it was chosen as the platform to test the project. Using OpenCV libraries we have implemented image processing mechanisms like RGB to grayscale conversion, erosion, dilation.

1.2 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy highlights code readability, and its syntax allows programmers to express concepts in less lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

1.3 Tesseract

Tesseract package contains an OCR engine - libtesseract and a command line program - tesseract. The lead developer is Ray Smith. Tesseract has Unicode (UTF-8) support and can recognize more than 100 languages "out of the box". It can be trained to recognize other languages. Tesseract supports various output formats: plain-text, hocr(html), pdf. The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England and Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migration from C to C++ in 1998. A lot of the code was written in C, and then some more was written in C++. Since then all the code has been converted to at least compile with a C++ compiler. Tesseract is available for Linux, Windows and Mac OS X, however, due to limited resources only Windows and Ubuntu are rigorously tested by developers.

1.4 Cloud Firestore

Cloud Firestore is a fast, fully managed, serverless, cloud-native NoSQL document database that simplifies storing, syncing, and querying data for your mobile, web, and IoT apps at global scale. Its client libraries provide live synchronization and offline support, while its security features and integrations with Firebase and Google Cloud Platform (GCP) accelerate building truly serverless apps.

2. RELATED WORK

Ming-Yee Chiu et al. proposed a method for counting the vehicles at the checkpoint from which the number of available parking spaces can be counted [1]. The counting is performed by installation of the induction loop sensors under the road surface. Although the usage of sensors was less costly, not easily affected by environmental conditions and it detects accurately however, it installation was difficult and cause damage to roads. It was also difficult to maintain it in case of malfunction [2]. Image processing is a hot research subject for quite a while and has scope for some developments in innovative applications. Image processing is required in all the major developing and advanced sectors of society like medical, security, engineering, entertainment, media and much more. Distinctive image pre-processing strategies important to accomplish higher exactness utilizing techniques like RGB to grey, blurring, thresholding and contouring [3] is significant. In certain areas enormous number of pictures are required to get to certain data. The Histogram method is executed to get statistical data from a image and categorize the image as poor or good [4]. It is likewise utilized for the normalization and equalizing the image. For mall, redundant or unwanted fragments in a image a two phase outline work of Histogram handling for separating and choice to expel undesirable little messages from a picture is utilized [5]. In License Plate Detection the text extraction is the significant objective in this way image processing turns into the most essential part preceding image to text conversion. The methodology in [6] uses the strategy for corner focuses for text extraction in document images. It has fixed parameters appointed for various kinds of pictures like handwritten, typewritten, skewed, etc and it is very quick. Paper [7] proposes the strategy for detecting license plate in an image taken from varied distances and different illuminations utilizing wavelet change and masking out potential license part. After the processing is finished then the processed image can be given to the Tesseract OCR Machine to change over the image into text utilizing command line interface [8].

3. ALGORITHM

We have used two algorithms to build the application of Smart Parking System

3.1 OpenCV

The still image file is a frame of the video file and after the code is executed, the still image shows and with the mouse, a quadrilateral would be constructed from its three input points. Coordinates of the input points are stored in a YAML file and then the letter 'q' shall be pressed to initialize the display of the video feed. The quadrilaterals created to zone each parking slot are outlined either red or green. Red means that the computer recognizes the parking space within the quadrilateral as occupied by a car and green means it is available for parking.

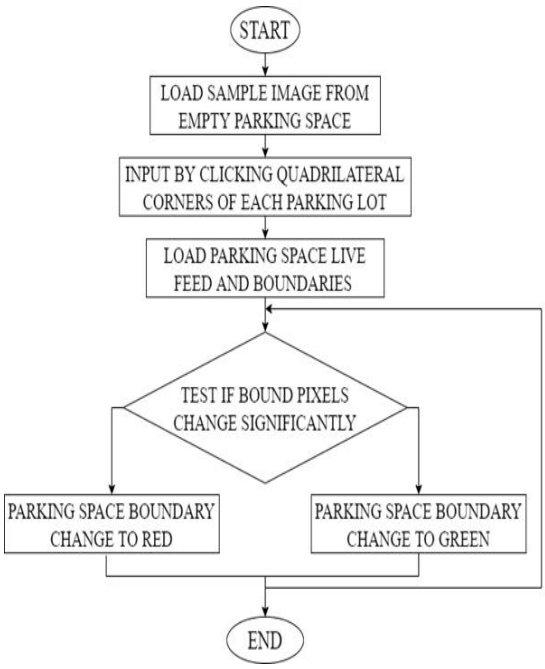

Fig -1: Flowchart of Empty slot Detection

3.2 Tesseract

When the car enters parking lot, we capture the image of the plate. We perform image processing methods on the captured image. We then find the contours of the plate and crop the image along those contours. The processed image is then fed to Tesseract algorithm for character recognition. The extracted image is then checked whether it exists in cloud or not. If the plate already exists in the cloud, we extract the information from it and produce bill based on time of stay. If the plate does not exist in cloud, we upload it with time of entry to the cloud.

Fig -2: Flowchart of Plate Recognition

4. METHODOLOGY

A car enters the parking lot and the parking lot is checked for empty slots. If there exists any then the number plate of the car is scanned using the first camera and stored in cloud and let into the parking lot by displaying the slots available. If there are no empty slots available, then the same will be displayed. When the car exits then the number plate is scanned again using the second camera and compared with the time of its arrival. A bill is generated based on time of stay of the vehicle.

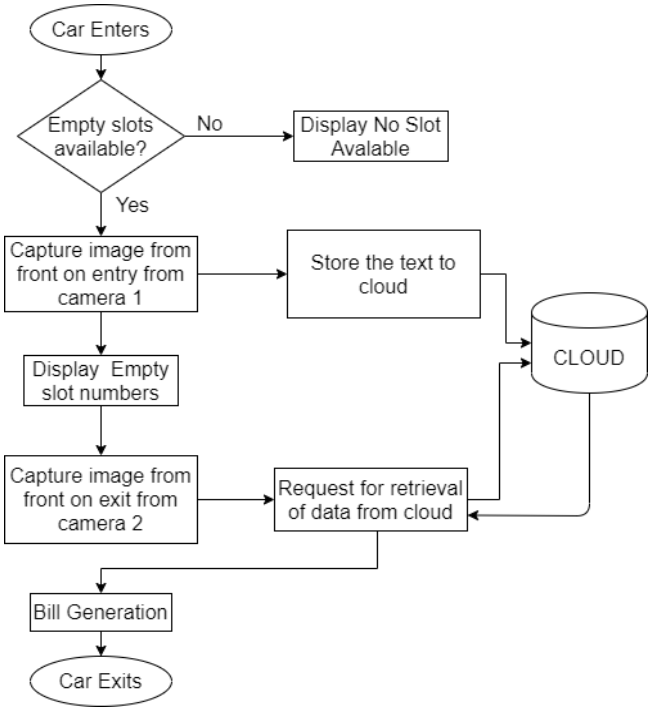

Fig -3: Flowchart of Smart Parking System

5. IMPLEMENTATION

Running the program, the Empty slot detection system has been exact in demonstrating whether a parking spot is occupied or not. Figures 4 and 5 show the condition of the parking structure and the building, masked with coloured rectangles.

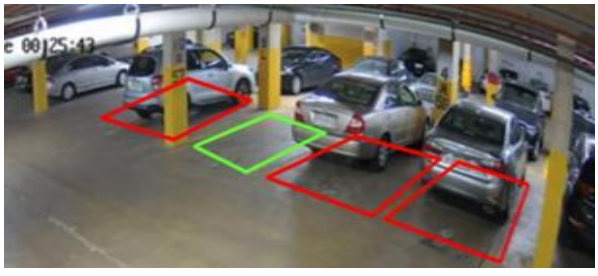

Fig -4: Parking garage where occupied space was marked red and unoccupied space was marked green

Fig -5: Building parking

Rectangle shape zones were set as input before the program was run. In the event that a vehicle is inside the given Rectangle shape zone, the pixel colour inside will change, and the average colour would likewise vary. This would incite the program to change the colour from green to red.

Be that as it may, there are a few situations where a parking spot may turn red regardless of whether a vehicle isn't really the one involving it. As appeared in Figure 6, any strong hindrance, for example, an individual walking by or a garage post that changes the in overall average pixel colour of the zone limited by square shapes might be considered by the algorithm as a vehicle along these lines provoking the layout to turn red.

Fig -6: Indoor parking lot with a garage post (video retrieved from: [10])

The quadrilateral regions are largely reliant on the initial still picture set apart during system initialization. On the off chance that the video outline changes position from the marked image, the conceal video feed yield would show uprooted square shape, as appeared in Figure 7. Consequently, the framework would work best in the event that the video feed is acquired through introduced CCTV surveillance cameras around the parking spot since they are stationary in place.

Fig -7: Basement parking

Figures 8 and 9 show the initial feed where the marked boxes are illustrated with green when the pavement is unfilled, or the available parking space is accessible, thereby, having an average pixel colour of grey.

Fig -8: Unoccupied indoor parking outlined as green by the program (video retrieved from: [7])

Fig -9: Occupied indoor parking outlined as red by the program (video retrieved from: [7])

At the point when the vehicles occupy a specific parking slot, the layout diverts quickly from green to red since the average colour has changed.

For the number plate detection, the image of the number plate is captured by pressing the button and OCR is processed to get the characters. The recognized number plate character is stored in the form of text in the Firestore cloud along with the time. This will be used again when the car leaves the parking lot and the bill will be generated.Fig.10 shows the image of the car which was taken at entrance

Fig -10: Number plate image captured at entrance

Fig.11 and Fig.12 shows the output of the characters detected from image of car taken at entrance and exit. It also shows the bill generated according to the time interval of parking.

```
Number is : MH12DE1433
MH12DE1433 added to Cloud


Process finished with exit code 0
```

Fig -11: Number plate detected at entrance

```
Number is : MH12DE1433
2020-05-12 15:04:11.443530
Checked In : 2020-05-12 15:04:11
Checked Out : 2020-05-12 15:06:19
You've stayed for 0:02:08
Your Bill = Rs.50

Process finished with exit code 0
```

Fig -12: Number plate detected at exit, along with bill generated.

6. CONCLUSIONS

Image Processing is very crucial for extracting any information from an image. In this study, a proposed plan for a smart parking system based on image processing has been effectively tried and run with a few videos taken from indoor parking garages. The system works precisely in deciding regardless of whether the parking slots are occupied or not by showing a red outline if a vehicle is inside or consuming a parking spot and afterward turns green when it is unoccupied. In the number plate detection, we first applied image processing algorithms to images and afterward those images were utilized in Tesseract software to acquire the text from the images. Different images have distinctive text styles, length, width and font, so different images require different levels of digital image processing techniques. Out of these image results for a single image the most appropriate image is then applied to Tesseract for obtaining the text from an image. Therefore, after digitally processing the image we have accomplished better and near to perfection outputs.

REFERENCES

[1] Ming-Yee Chiu; Depommier, R.; Spindler, T.; , "An embedded realtime vision system for 24-hour indoor/outdoor car-counting applications," Pattern Recognition, 2004.

[2] Zhang Bin; Jiang Dalin; Wang Fang; Wan Tingting; , "A design of parking space detector based on video image," Electronic Measurement & Instruments, 2009.

[3] F. Zhou, and Q. Li, "Parking guidance system based on ZigBee and geomagnetic sensor technology," 13th International Symposium on Distributed Computing and Applications to Business Engineering and Science (DCABES) 2014, pp. 268-271.

[4] Y. Zheng, S. Rajasegarar, and C. Leckie, "Parking availability prediction for sensor-enabled car parks in smart cities," IEEE 10th International Conference on Intelligent Sensors Sensor Networks and Information Processing (ISSNIP) 2015, pp. 1-6.

[5] A. Khanna, and R. Anand, "IoT based smart parking system," 2016 International Conference on Internet-of-Things and Applications (IOTA), Pune, 2016, pp. 266-270.

[6] Z. Ji, I. Ganchev, M. O'droma, and X. Zhang, "A cloud-based intelligent car parking services for smart cities," General Assembly and Scientific Symposium (URSI GASS) 2014 XXXIth URSI, pp. 1- 4.

[7] K. Cheng, "Indoor Parking DAHUA," 2018. [Online]. Available: https://www.youtube.com [Accessed 2018].

[8] KETOSI, "[KETOSI] Audi R8 V10 Parking Lot Accident CCTV," 2013. [Online]. Available: https://www.youtube.com [Accessed 2018].