Real Time Parking Space Finder

Mrs. Jayanthi S[1], Andrew shelton[2], Aravindhan[3], Deepak[4]
[2,3,4]*UG Student, Department of CSE, Agni College of Technology, Chennai*
[1]*Assistant Professor, Department of CSE, Agni College of Technology, Chennai*

Abstract **- The aim of OpenCV is to help the computer to manipulate the content of an image and do some specific tasks over it. OpenCV is a library of programming functions mainly used for image processing. We can solve many real time problems using image processing applications. The rise in the field of OpenCV makes way for many new inventions. In this paper, we can implement this concept to track whether the parking space is occupied or not using OpenCV to detect vehicles. Detection of those vehicles is mainly based on motion detection, Whereas the module is implemented in Raspberry Pi single board computer to achieve IoT automation. It regularly tracks the parking space and compares with the empty parking lot coordinate image. If any free space available it will be automatically sent to the remote database, which is then retrieved by the user while accessing the website or mobile app and finding the available spaces.**

Index Terms **- OpenCV, Laplacian Transformation, IoT automation, web tech**

I.INTRODUCTION

Due to increase in traffic and Vehicles manufacturing it would be hard to find a suitable parking space in a parking lot. The concept of Internet of Things with Image Processing has gained enough interest in the field of Agriculture and Industries. Which have achieved a certain degree of success. However, the combination of both these techniques is not existent so far in the real-world problems other than these fields. The combination of the techniques can be implemented to track the free space of the parking lot with efficient setup. Which will also lower the maintenance and service costs while compared to the traditional sensor setup. It also achieves some security from the vulnerable physical parameters of the environment.

II.EXISTING METHODS

The Existing methods involves traditional detecting hardware such as motion detection sensors, proximity sensors which are programmed via a microcontroller coupled with a microprocessor. They are grouped together using the microcontroller, which senses the surrounding space around it to find the availability of the parking space. This kind of systems is in work for a long period of time which also includes high level of setup and maintenance cost. More importantly they are highly vulnerable to the physical factors of the environment. These sensors transmit the precepted values to the microcontroller which is then transferred to the microprocessor and some algorithm takes place on the received values to find the free spaces. Such methods can fail if one of the sensors wear out or failed to work as required which creates fault values.

III.PROPOSED SYSTEM

The proposed system planned to use Motion detection to find the available parking spaces using OpenCV. Motion detection will detect the changes in the real time input by movement of vehicles. The proposed system will overcome the limitation of the existing system which was having a large setup and maintenance cost and does not have dynamic updates. The proposed system will automatically update the status of the parking lots in the database and thereby showing it in the UI in real time.

IV.MARKING COORDINATES

First the system provider needs to mark the coordinates of the parking spaces in the particular parking lot which needs to be done manually for accuracy. The marking process will be carried out using OpenCV paint function in a window. The process needs an empty image of the particular parking lot showing all the spaces in which the marking has to be made. Once the marking has been

completed the marked values will be recorded over a yaml file and it will be used in the next process for detection of the vehicles in the marked spot.

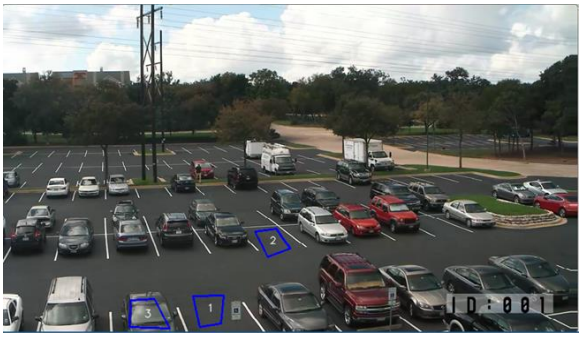


Fig 1. Marking of parking spaces

V.MOTION DETECTION

The motion detection process does multiple process on the video by taking every frame. The main concept is to find the mean value for the marked coordinates using Laplacian transformation. Here the marked coordinate will be considered as a square where the area of the square is identified and selected after that, a mean value of the dark pixels is compared with the given Laplacian threshold and a final decision is made as either the space is free or not.

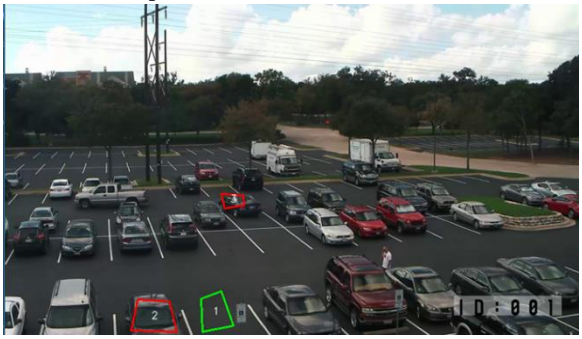


Fig 2. Detection of available free spaces

VI.DATA BASE UPDATE AND RETRIEVAL

After the identification of the available spaces, they are updated in the MySQL database furthermore this will be kept on updating until the system lives for live tracking purposes. These information from the database will be retrieved by the program that runs on the website accordingly and displays them on the UI of the mobile app and website.

VII.GAUSSIAN BLUR

In this approach, instead of a box filter consisting of equal filter coefficients, a Gaussian kernel is used. It is done with the function,

cv2.GaussianBlur(img,(sigX,sigY),kernel)

We should specify the width and height of the kernel which should be positive and odd. We also should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as equal to sigmaX. If both are given as zeros, they are calculated from the kernel size. Gaussian filtering is highly effective in removing Gaussian noise from the image.

VIII.BGR TO GRAY CONVERSION

For color conversion, we use the function cv2.cvtColor(input_image, flag) where flag determines the type of conversion For BGR -> Gray conversion we use the flags cv2.COLOR_BGR2GRAY. Similarly, for BGR -> HSV, we use the flag cv2.COLOR_BGR2HSV.

IX.LAPLACIAN

It calculates the Laplacian of the image given by the relation where each derivative is found using Sobel derivatives. If ksize = 1, then following kernel is used for filtering:

$$\Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$$

Here it is implemented using cv2.Laplacian().

X.SOBEL OPERATOR

Sobel was based on the fact that in the edge area, the pixel intensity shows a “jump” or a high variation of intensity. Getting the first derivative of the intensity, we can observe that an edge is characterized by a maximum, as it can be seen in the figure:

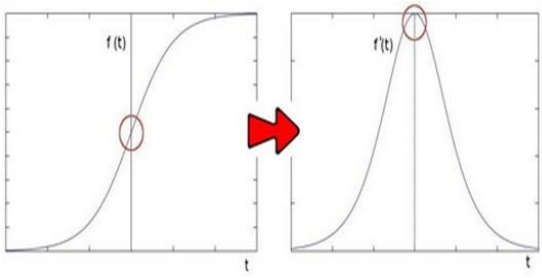


Fig 3. Soble Transformation after taking differentiation

XI.LAPLACE OPERATOR

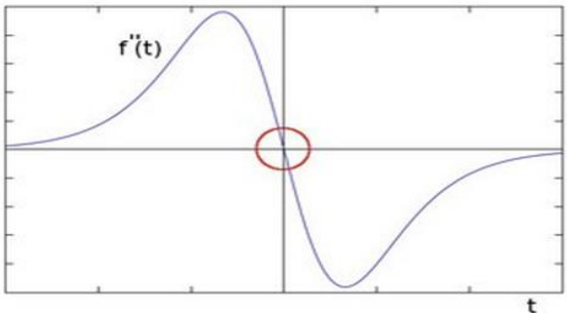
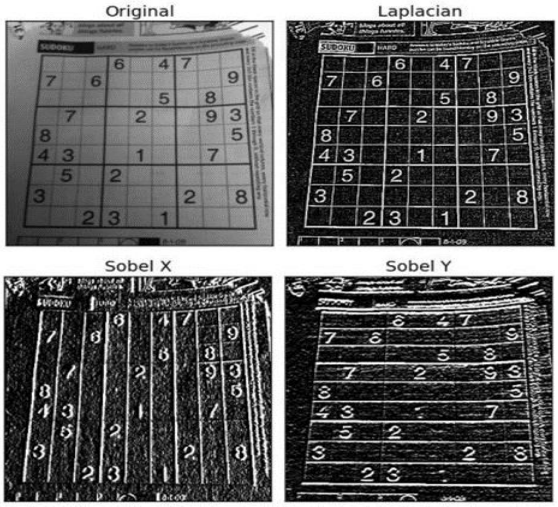


Fig 4. Laplacian Transformation after taking differentiation on Sobel

XII.EXAMPLE OF LAPLACIAN TRANSFORMATION



XIII.PROPOSED FORMULA

Mean of Grouped Data:

$$\bar{x} = \frac{\sum fx}{n}$$

where: $\bar{x}$ = mean
f = frequency of each class
x = laplacian matrix * mask[current index]
n = total frequency
$\sum fx$ = sum of the producst of mid − interval values and their corresponding frequency

status = mean < laplacian

XIV.CONCLUSION

The system will make use of the limited resources to find the available parking spaces with efficiency compared to other traditional methods and will also enables the parking lots interlinked to one another using IoT within a city or even a metropolitan area as required. The system will work as stand-alone module which does not require any centralized server or any other manual operations. This ensures the working of other modules in another parking lot to work even when one of the interlinked chains of system fails. The accuracy of the detection is preserved by manual marking of coordinates for once and will be used for the rest of the system life. It will save the time for the user to find the available spaces in peak hours lively. The user will be notified with a user-friendly Website and Mobile App.

REFERENCES

[1] Jan Erik Solem “Programming Computer Vision with Python: Tools and Algorithms for Analyzing Images”,2012.
[2] Saurabh Kapur “Computer Vision with Python 3”,2017.
[3] Urakawa Hajime “Spectral Geometry of The Laplacian: Spectral Analysis and Differential Geometry of the Laplacian”,2017.
[4] Robert Hummel “Deblurring Gaussian Blur”,2018.
[5] Prateek Joshi “OpenCV with Python by Example”2015.
[6] Rico, J., Sancho, J., Cendon, B., & Camus, M. (2013, March). Parking easier by using context information of a smart city: Enabling fast search and management of parking resources. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on (pp. 1380-1385). IEEE.
[7] 2017- A navigation and reservation based smart parking platform using genetic optimization for smart cities-lhan Aydin; Mehmet Karakose; Ebru Karakose.
[8] Parwekar, P. (2011, September). From Internet of Things towards cloud of things. In Computer and Communication Technology (ICCCT), 2011 2nd International Conference on (pp. 329-333). IEEE.

[9] 2015- An approach to IOT based car parking and reservation system on cloud Vaibhav Hans, Parminder Singh Sethi, Jatin Kinra.

[10] 2016- Novel vehicle booking system using IOT_S. Vidhya Sagar; B. Balakiruthiga; A. Sivanesh Kumar