Scientific Journal of Silesian University of Technology. Series Transport
Zeszyty Naukowe Politechniki Śląskiej. Seria Transport

Volume 106 **2020**

p-ISSN: 0209-3324

e-ISSN: 2450-1549

DOI: https://doi.org/10.20858/sjsutst.2020.106.17

Journal homepage: http://sjsutst.polsl.pl

Article citation information:
Trivedi, J., Devi, M.S., Dhara, D. Canny edge detection based real-time intelligent parking management system. *Scientific Journal of Silesian University of Technology. Series Transport*. 2020, **106**, 197-208. ISSN: 0209-3324.
DOI: https://doi.org/10.20858/sjsutst.2020.106.17.

Janak TRIVEDI[1], Mandalapu Sarada DEVI[2], Dave DHARA[3]

CANNY EDGE DETECTION BASED REAL-TIME INTELLIGENT PARKING MANAGEMENT SYSTEM

Summary. Real-time traffic monitoring and parking are very important aspects for a better social and economic system. Python-based Intelligent Parking Management System (IPMS) module using a USB camera and a canny edge detection method was developed. The current situation of real-time parking slot was simultaneously checked, both online and via a mobile application, with a message of Parking "Available" or "Not available" for 10 parking slots. In addition, at the time entering in parking module, gate open and at the time of exit parking module, the gate closes automatically using servomotor and sensors. Results are displayed in figures with the proposed method flow chart.

Keywords: Raspberry Pi, parking, edge detection, Python, real-time, sensors

1. INTRODUCTION

Parking problems in day to day life is the current scenario in most cities. The lack of real-time information for slots/parking positions in a particular location motivated this work on

[1] Faculty of Electronics & Communication Engineering Department, Gujarat Technological University, Government Engineering College Bhavnagar-364002, Gujarat, India. Email: Trivedi_janak2611@yahoo.com
[2] Principal, Ahmedabad Institute of Technolog-380060, Gujarat Technological University, Gujarat, India. Email: saradadevim1@gmail.com
[3] Faculty of Electronics & Communication Engineering Department, Gujarat Technological University, Government Engineering College Bhavnagar-364002, Gujarat, India. Email: dave.dhara24888@gmail.com

real-time parking management module using Raspberry Pi for live video. The primary goal of this framework is to develop a small module using open source software, hence, python was used for this establishment. Current research demands hardware-based simulation results and the Internet of Things (IoT). Therefore, we have developed a real-time parking facility, accessible to the client on a portable mobile application.

Live-stream is captured using a USB camera. Real-time status of parking is displayed on the LCD screen, which was present in the parking module and slots availability can be found via the mobile application. The camera is mounted at a corner to take snapshots of the object and send pictures to the processing or controller unit, in this case, it is a raspberry pi. The android module is connected to the raspberry pi through the Wi-Fi. When user enters in parking area, he/she sees the number of available parking slots and for further information, he/she simply sends message "Slots" from the mobile android application to the "Parking System" and in response, user gets the information about the parking slots and real-time image of the parking area.

In this article, the literature survey about hardware module-based parking system or research scenario in IoT is explained in section 2. The Canny edge detection method is explained in section 3. In section 4 is explained the proposed method with a flowchart using hardware description and also show results. The final section deals with discussion and conclusion with future scope from this work.

2. LITERATURE SURVEY

The Internet of Things (IoT) is an added essence in the combination of hardware and software computing. Intelligent Transportation System (ITS) is one of the main aspects of smart city development as parking is a part of ITS. Firstly, in the computer vision field, we have to detect edges from the image to perform a further task. John Francis Canny in 1986 demonstrated canny edge detection with numerical mathematical formation. A case study for vehicle parking for outdoor and indoor conditions using GNSS and ultra-wideband technology with the help of a state-of-the-art method is explained in [4]. Attempts were made at overcoming the limitations of static and dynamic obstacle in this article using IoT based real-time traffic management module in our proposed method with the help of Canny edge detection and adaptive thresholding.

IoT based design and implementation prototype of a smart city for Smart Street Cosenza (SSC) (Italy) was presented in earlier [7] using iSapeins edge-based platform. In this article, a future scope traffic control system is planned to be designed, and part of the traffic control system, real-time traffic management is implemented here with results.

GPS-based highway toll collection system, using a combination of hardware (Raspberry Pi, microcontroller, GPS and LCD module, Wi-Fi adapter, speaker, wi-fi router) and software (SQL database, cloud server) for reducing traffic congestions is explained in [16]. The automatic time-delay adjustment system is also included for improving accuracy in the final result.

Smart Park system for real-time parking availability using mobile sensor and wi-fi, with the help of random forest-based approach was explained earlier [11]. Improved quality of service as a part of an Intelligent Transportation System (ITS) with the help of providing real-time public bus travel information system via a mobile app is explained in [1]. Vehicle detection using colour features, vehicle tracking using Kalman filter and vehicle counting with the help of Raspberry Pi 3, open CV and C++ is explained in [3]. Survey of machine

learning-based different IoT application in traffic engineering, security with challenges and open issues is explained in [5,9]. Face and fatigue detection using Haar descriptors with the help of Raspberry Pi board, USB camera, Open CV libraries, and Python is explained in [10]. Hybrid smart parking module, with a survey of the use of different sensors under smart city development using IoT, is explained in [2].

3. CANNY EDGE DETECTION

Canny edge detection is a customised algorithm to find edges and was described by John Francis Canny in 1986 [6]. Localisation and specification of detection-based, mathematical approach for finding step edges with the help of adaptive thresholding method. First, in image processing, the Gaussian Kernel with different size is applied. Gaussian Kernel is defined in 1-D, 2-D or N-D as mentioned in equation 1 - 3 below [8]

$$G_{1d}(x,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{-x^2}{2\sigma^2}} \quad (1)$$

$$G_{2d}(x,\mathrm{y},\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{-x^2+y^2}{2\sigma^2}} \quad (2)$$

$$G_{Nd}(x,\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{-|x|^2}{2\sigma^2}} \quad (3)$$

σ is a standard deviation and σ2 is a variance in the statistics.

3.1 Filtering operation

Filtering in Camera, WhatsApp, Instagram, and different apps is a combination of many low-level processing of various types like contrast changes, colour changes, blurs, etc. In day to day life Gaussian filters, edge detection takes some images as input, process it and give us the desired output. Kernel convolution is used for edge detection and/or filtering operation.

3.1.1 Kernel convolution

A simple convolution is defined as mentioned in the equation below,

$$y(x,y) = h(x,y)*g(g,y) \quad (4)$$

Kernel convolution is a simple process with taking a small grid of numbers and pass them over whole images and transforming based on what numbers they are. With the use of different numbers in the kernel, we are able to perform the various operation, like blur, edge detection, sharpen, unsharpened, etc.

5	5	5	10	10
5	5	5	10	10
5	5	5	10	10
5	5	5	10	10
5	5	5	10	10

Fig. 1. Pixel information of the image

1	1	1
1	1	1
1	1	1

Fig. 2. A mean kernel

Generally, a kernel is quite small, even smaller than the image. In this process, we moved the kernel along with the image so that the pixel is in the centre. Looking at the kernel 3 X 3, with all values are 1, work as simple mean filtering operation. Multiplying kernel (Fig. 2) with the kernel (Fig. 1) gets Fig. 3.

5	5	10
5	5	10
5	5	10

Fig. 3. This is 3 X 3 matrix, selected from image (Fig. 1), for kernel convolution

After multiplying, normalising by dividing the total value of the kernel. Here, the total value of kernel is 1+1+1+1+1+1+1+1+1=9. Therefore, it is like averaging operation. Answer after kernel convolution is = Addition of all elements / Normalised value. Therefore 5 is now replaced by, (60/9 = 6.67). Continued this way to get the final output image. Sometimes blurs are used to remove noise from images before processing it. We can achieve different effects by using different values of the kernel. Blurring around the edges is ever so slightly less than the blurring around the image. In the last edge pixel in a megapixel image probably will not make any difference. We have used the Gaussian kernel in this illustration. Gaussian blur is commonly used in image processing because it is the more controlled and edge-preserving kernel. A normal distribution is a bell curve, and the standard deviation is essentially the average distance from the mean of all the points. Large standard deviation (σ) will have a large curve, and it should be symmetrical. Small standard deviation (σ) have a tight bell curve. Example of small Gaussian blur, with 3 X 3 Kernel is shown in Fig. 4.

1	2	1
2	4	2
1	2	1

Fig. 4. Example of Gaussian Kernel with 3 X 3 matrix

Difference between Figs. 3 and 4 kernels, with priority given to the middle values in both kernels. The further away get from the pixel of interest, the less interest in the combined average. The fact that it is not blurring too much makes for an interesting point. For an edge sharp change in intensity values are less variable compared to mean blur. In the image processing method, we have to increase the size of the kernel as the standard deviation (σ) or radius of the Gaussian function increase. After applying the Gaussian kernel to the same in Fig. 2, 5 is replace with, (5+10+10+10+20+20+5+10+10)/16 = 6.25, whereas in earlier case that value was 6.67. So, 6.25 is a nearer value compare to 6.67, predicts better result with changes in edges using the Gaussian kernel.

3.2 Edge Detection

Edge detection is simply a case of trying to find the regions in an image, where we have sharp changes in intensity or a sharp change in colour. A high value indicates sharp changes and low values indicate a shallow change. A very common operator is the Sobel operator,

which was derived by Irwin Sobel and Gary Feldman in 1968 [6], which is an approximation to a derivation of an image with separate in x and y direction, as shown in Fig. 5.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 5. Gradient in xand y directions

If, we are operated kernel convolution (Fig. 1) with Gx from Fig. 5, -5+0+10-10+0+20-5+0+10 = 20, and now if all values are same in Fig. 2, then the answer is 0, with one side dark or bright. Kernel convolution (Fig. 2) with Gy from Fig. 5, -5-10-10+0+0+0+5+10+10=0. So, gradient value and orientation of edge is given by equations 5 and 6,

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{5}$$

$$\text{Angle} = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{6}$$

As of equation magnitude is always positive and value of zero will be a consistent colour. If the gradient in the x-direction is big and gradient in the y-direction is small, then a moderate gradient should be obtained. Also, if the gradient in both directions is big, then a large gradient value should be obtained. This gradient value represents, how big the edge is in a particular location. We can also calculate the angle of the edge using equation 6. These equations help find structures or objects. First, we need to convert our colour image to greyscale image, taking only change of intensity and then use Gaussian filter.

The canny edge detector essentially takes a Sobel operator and makes it step better. The input of the canny operator is the output of the Sobel operator. Thinning all the images, so they are 1 pixel wide. Canny edge operator is first finding the edges then using a process called hysteresis thresholding, which is two-level thresholds. Every pixel first finds local maxima, that is, bigger than its neighbours after calculation of gradient and orientation. This process should be completed over the entire image. The second stage to remove the edges that even though not maximum, or weaker response. Create an image with dominant edges and preserves only dominate edges, and this process is done by hysteresis thresholding. Two-level thresholds, upper-level T1 and lower- level T2 decides whether objects edge in, edges out or in if connected. Adjust upper and lower level threshold of hysteresis thresholding, for controlling the output of canny edge detection.

4. PROPOSED METHOD

First, select Region of Interest (ROI) that include only white background. Apply Gaussian filter with sigma (σ) value of 0.3. Then apply a canny edge detector with defined lower and upper limit value to a set statistical parameter. Detect edges in (x, y) location, and then create a counter for object availability in the parking area, which is already shown in the

main flowchart in the Figs. 6, 7 and 8 represented as the separate process for finding the outcome.

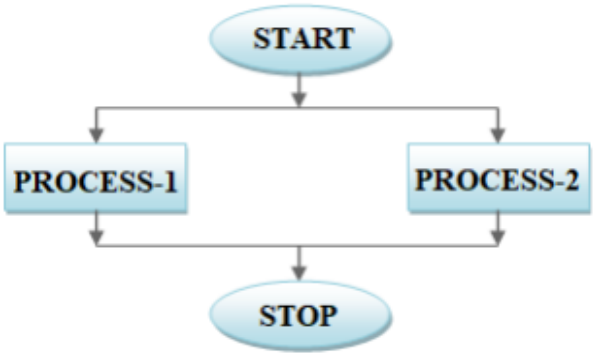

Fig. 6. Main flowchart.

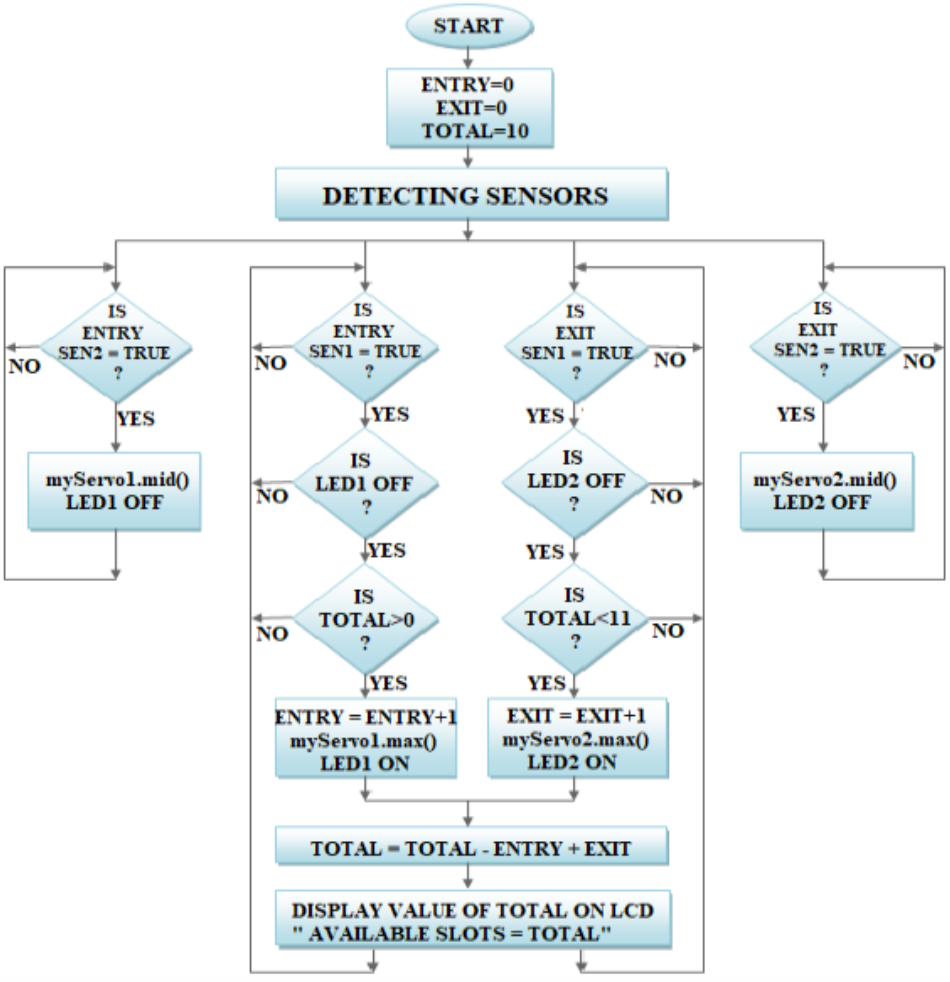

Fig. 7. Process-1

4.1 Process-1

Process-1 is related to the entry and exit module. This process is a continuous process, and sensors continuously detect the presence or absence of an object. During start of the procedure entry, exit and a total number of slots values are 0,0 and 10, respectively. Now, if sensor-1 entry is True, that is, a car is detected, it will check the status of the LED. If the LED is off that means it is false, thereafter, it will check the total predefined number whether it is greater than zero or not. If it is zero then there is no available slot in the parking area. When a condition is true then the value of the entry is increased by one, the gate opens and the LED comes on. Simultaneously, the number of available slots is updated in a display. When the sensor-2 entry is true, the gate closes and the LED turns off. Similar kind of process is done at exit mode. The main difference between entry and exit mode is that when an exit is detected, a decrease in the count of numbers of available slots and accordingly update the display.

4.2 Process-2

Process-2 is about detecting a real-time image. When a Raspberry Pi gets the command “Slots” from the Android module, it initially captures the real-time image of the parking area.

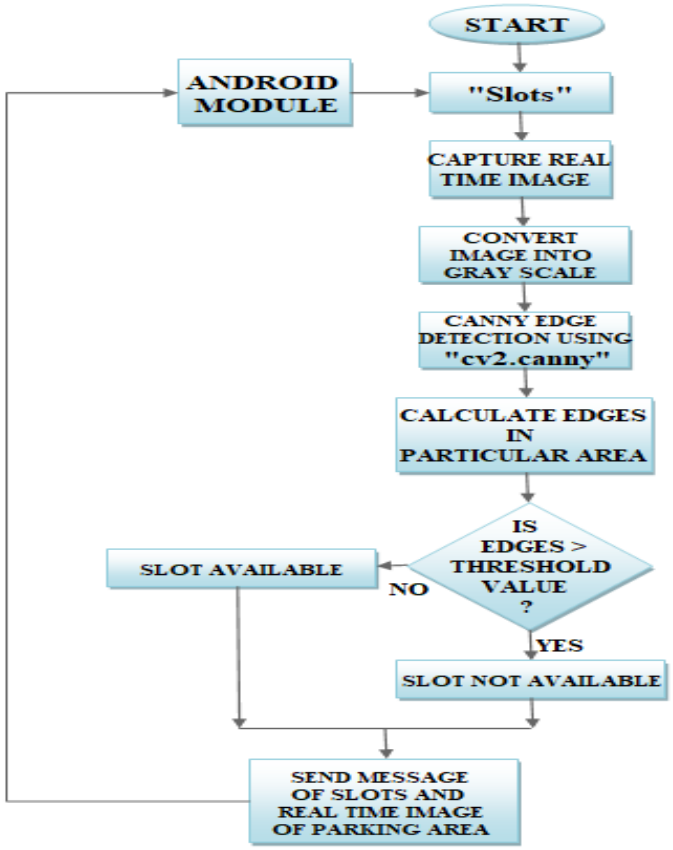

Fig. 8. Process-2

4.3 Pseudocode

Start
Video Read
Image conversion (RGB to Grey)
Canny-edge-detection for object detection
Statistics calculation to parameter
Apply filtering operation
Hysteresis thresholding
End

Parking slots "Available" or "Not available" can also be detected via an android mobile with internet facility. That means if I am currently situated at location A, and I have to go to location B, with known parking small module, for example, 10 number of parking slot, I can easily check using a network connectivity based android module device. Here, Raspberry Pi has 40 pins for input/output and every I/O pin gives 3.3V output. It has also two separate pins of 5V [14].

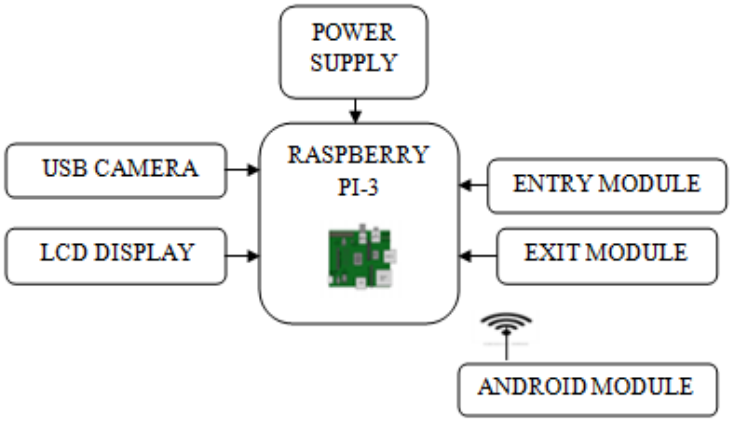

Fig 9. System Architecture

5. RESULT

Apply proposed method to parking module with 10 toy vehicles, USB camera, raspberry pi, and an android module for observing/monitoring real-time parking slots availability by the client and the LCD device for displaying same results at the monitoring side. Two servo motors used for opening and closing parking door, 4 -IR obstacle detection sensors used for detection of a real-time object.

Figure 10 is shown with small parking module as a part of the Intelligent Parking Management System (IPMS), with entry and exit slots and hardware connection. Figure 11 is checked parking status with a real-time image from the parking module represented in Figure 10 with the help of the android module. Here, different colour cars are used for monitoring the parking management system. Parking slots are reserves with black spot, for a fixed position of parking. If a rectangular black spot is visible, then parking slots are available otherwise parking slots are not available. Figures 11 and 12 are shown in a combined result: with

the LCD screen display at parking module and in an android mobile application with real-time image, for full parking and seven available slots, respectively.

In Figure 12, white, yellow, green, red colour car with different sizes and shapes, placed in parking modules and after applying the proposed method, it accurately displays result as message “Parking Full” on the LCD screen, for the server and for the client, who are present at that particular location. Moreso, at the same time, for clients who are not present at a particular location of a parking site. After some time, he/she will reach the specified location of parking and establish their parking position via an android module in real-time.

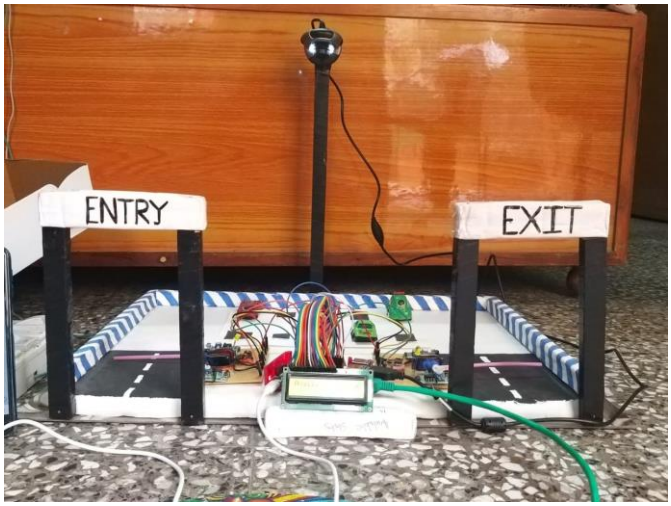

Fig. 10. Small parking module for 10 vehicles parking position

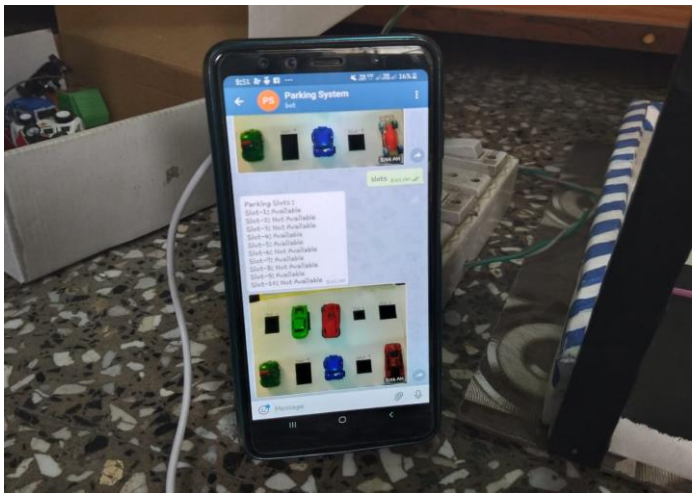

Fig. 11. Checking parking status in android mobile with a real-time image from the parking module

6. DISCUSSION

In this article, we have presented a real-time parking management system module for 10 toy car using the Raspberry Pi hardware. Different sizes, shapes, and colours were not affected by the proposed method. Real-time parking position monitors using the LCD screen

and any android mobile device using message application at the same time. Raspberry Pi module and the android device were connected using wi-fi connection. Entry module and exit module consists of two sensors, one servo motor and Light Emitting Diode (LED) each. A sensor first detects the car, gives the input to the motor and the gate opens, and the LED blinks the signals of car detection. Afterwards, the second sensor detects the car, gives input to the motor and gate closes with the LED going off. The exit gate observes a similar procedure as well. This proposed method not only solves the traffic congestion problem but it is also helpful for reducing noise, which negatively affects human life. The environment and the economy benefit because less fuel and time is required for parking.

Fig. 12. Combined result: with LCD screen display-parking module and in an android mobile application with a real-time image for the Full Parking

6. CONCLUSION

In this article, we represented an intelligent parking monitoring system module using the canny edge detection method, hysteresis thresholding for 10 parking slots. Real-time parking information available in an android module with slots availability as well on the LCD screen, available at the parking location. Here, proposed method results are not at variance with the different size and shape of four-wheeler vehicles. Advantages of implementing this method in real-time is to reduce traffic congestions, parking-related problems via intelligent parking management systems. Real-time parking results are varied accordingly by the camera position, hence, a fixed camera position with almost 90° inclinations are positioned for only 10 cars. Real-time parking booking management, parking price policy, in an unknown place, finding parking position well in advance are the different applications possible using this proposed method.

In the future, we hope to develop a standalone device for a real-time intelligent parking management system for smart city development.

References

1. Akande N.O., et al. 2018. “Improving the quality of service in public road transportation using real time travel information system”. *World Review of Intermodal Transportation Research* 7(1): 57-79. DOI: 10.1504/WRITR.2018.089529.
2. Al-Turjman F., A. Malekloo. 2019. “Smart parking in IoT-enabled cities: A survey”. *Sustainable Cities and Society* 49. DOI: 10.1016/j.scs.2019.101608.
3. Anandhalli M., V.P. Baligar. 2018. “A novel approach in real-time vehicle detection and tracking using Raspberry Pi”. *Alexandria Engineering Journal* 57(3): 1597-1607. DOI: 10.1016/j.aej.2017.06.008.
4. Antoniou, C., et al. 2018. “A framework for risk reduction for indoor parking facilities under constraints using positioning technologies”. *International Journal of Disaster Risk Reduction* 31: 1166-1176. DOI: 10.1016/j.ijdrr.2017.09.032.
5. Asghari P., A.M. Rahmani, H.H.S. Javadi. 2019. “Internet of Things applications: A systematic review”. *Computer Networks* 148: 241-261. DOI: 10.1016/j.comnet.2018.12.008.
6. Canny J. 1986. “A Computational Approach to Edge Detection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(6): 679-698. DOI: 10.1109/TPAMI.1986.4767851.
7. Cicirelli F., et al. 2017. “An edge-based platform for dynamic Smart City applications”. *Future Generation Computer Systems* 76: 106-118. DOI: 10.1016/j.future.2017.05.034.
8. Course-Hero: 3. The Gaussian Kernel. Available at: https://www.coursehero.com/file/12384739/diffusiongaussiankernel/.
9. Cui L., et al. 2018. “A survey on application of machine learning for Internet of Things”. *International Journal of Machine Learning and Cybernetics* 9(8): 1399-1417. DOI: 10.1007/s13042-018-0834-5.
10. Isaza C., et al. 2019. “Dynamic set point model for driver alert state using digital image processing”. *Multimedia Tools and Applications. Multimedia Tools and Applications* 78(14): 19543-19563. DOI: 10.1007/s11042-019-7218-z.
11. Krieg J.G., et al. 2018. “Unlocking the smartphone’s sensors for smart city parking”. *Pervasive and Mobile Computing* 43: 78-95. DOI: 10.1016/j.pmcj.2017.12.002.
12. Muñuzuri Jesús, André Alho, João de Abreu e Silva. 2019. “Evaluating freight loadind/unloading parking zones characteristics, usage and performance in Southern Europe”. *European Transport \ Trasporti Europei* 73(5). ISSN: 1825-3997.
13. Patkar Manish, Ashish Dhamaniya. 2019. “Effect of on-street parking on effective carriageway width and capacity of urban arterial roads in India”. *European Transport \ Trasporti Europei* 73(1). ISSN: 1825-3997.
14. Raspberry Pi: Raspberry Pi 3 Model B. Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.
15. Sobel Irwin, Gary Feldman. 1986. “A 3×3 isotropic gradient operator for image processing”. Stanford Artificial Intelligence Laboratory (SAIL).
16. Tan J.Y., et al. 2017. “GPS-based highway toll collection system: Novel design and operation”. *Cogent Engineering* 4(1): p 1-10. DOI: 10.1080/23311916.2017.1326199.

Received 10.11.2019; accepted in revised form 20.01.2020