**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY**



(An Autonomous Institution)  
(Approved by AICTE and Affiliated to Anna University, Chennai) ACCREDITED BY NAAC WITH “A” GRADE

**PARKING SPACE DETECTION**

**A PROJECT REPORT**

***Submitted by***

D.GOKULPANDY (17EUIT048)

R.ASHWIN (17EUIT018)

C.AJIT (17EUIT004)

A.AJITH PRASANNA (17EUIT005)

***In partial fulfillment for the award of the degree***

***Of***

**BACHELOR OF TECHNOLOGY**

In

**INFORMATION TECHNOLOGY**

MARCH 2021

**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY**   
(An Autonomous Institution)  
(Approved by AICTE and Affiliated to Anna University, Chennai) ACCREDITED BY NAAC WITH “A” GRADE



**BONAFIDE CERTIFICATE**

Certified that this project report

**“PARKING SPACE DETECTION”**

Is the bonafide work of

D.GOKULPANDY (17EUIT048)

R.ASHWIN (17EUIT018)

C.AJIT (17EUIT004)

A.AJITH PRASANNA (17EUIT005)

Who carried out the project work under my supervision.

[NAGENDRAN K] [Dr.S.SUSILA]  
**SUPERVISOR** **PROFESSOR & HEAD**  
Assistant Professor Information Technology,  
Information Technology Sri Krishna College of Engg & Tech  
Sri Krishna College of Engg & Tech Coimbatore - 641008.  
Coimbatore - 641008.

This project report is submitted for the autonomous project viva-voice examination held on.....................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and gratitude to Principal, Sri Krishna College of Engineering and Technology for her kind patronage.

We profusely thank, Professor & HOD, Department of Information Technology, who has greatly helped in the success of the project, by providing us with the necessary equipment and facilities required.

We convey our regards to our tutor, Department of Information Technology Sri Krishna College of Engineering and Technology, for her guidance and encouragement, all through the course of the project.

We express our sincere thanks to our Faculty Guide, Professor & The year co–Ordinator, Department of Information Technology, Sri Krishna College of Engineering and Technology, who was always ready to help us and played a major role in the completion of project.

We would like to express our gratitude to our Project Committee Member, Department of Information Technology, who has given as many ideas for doing this project.

Finally, we thank all the members of the Information Technology Department, Sri Krishna College of Engineering and Technology, who have been a source of greater help in our venture.

**ABSTRACT**

The main scope of the presented Project was the development of an innovative product for the management of city parking lots. Our application will ensure the implementation of the Smart City concept by using computer vision and communication platforms, which enable the development of new integrated digital services. The use of video cameras could simplify and lower the costs of parking lot controls. In the aim of parking space detection, an aggregated decision was proposed, employing various metrics, computed over a sliding window interval provided by the camera. The history created over 20 images provides an adaptive method for background and accurate detection. The system has shown high robustness in two benchmarks, achieving a recognition rate higher than 93%. Looking through a reasonable parking spot in a populated metropolitan city is amazingly hard for drivers. Genuine gridlock may happen because of the inaccessible parking spot. Programmed shrewd stopping framework is arising field and pulled in PC vision scientists to contribute in this field of innovation. In this paper, we have introduced a dream-based shrewd stopping system to help the drivers productively finding appropriate stopping openings and hold them. At first, we have divided the stopping territory into blocks utilizing alignment. At that point, order each square to recognize the vehicle and personal the driver about the situation with leaving either held or free. Conceivably, the exhibition exactness of the suggested framework is higher than cutting edge equipment arrangements, approving the matchless quality of the proposed system.

**CHAPTER 1**

**INTRODUCTION**

One of the main components of the parking management system is the occupancy detection. On a large scale implementation, the system requirements will be easy to use, quick, and within budget. The objective of the current solution is to be very accurate, as it is generally used in the context of paid-for parking lots. As a result, the vast majority of the existing solutions are sensor-based devices (e.g., proximity sensors) associated with each available parking space. The large number of sensors needed with the associated network system (possibly wireless), the electrical power supply required (possibly using rechargeable batteries), and the complexity of installation (in-road implantation), together with the associated maintenance activities raises the cost of such a solution, making it prohibitive when considering its use for a whole city or an extensive area. We are proposing a method for detecting parking spaces occupancy using fixed cameras. This type of approach involves processing video feeds and recognizing the presence of vehicles by using computer vision methods.

**1.1 OVERVIEW**

The proposed method comes with significant benefits over the classic detection methods: significantly lower costs associated with the initial implementation, minimal costs associated with scaling, easy reconfiguration of an existing parking lot, and the possibility to also use and record the video feeds for surveillance and security purposes. Much of the current research focuses on identifying the available and occupied parking spaces based on image processing techniques. There are, however, several approaches to the detection of available/occupied parking spaces, as in the paper, suggesting a solution to provide robust parking performance and to address the occlusion problem between spaces. In order to extract the parking model, a model based on street surfaces is suggested, being a less occlusive one. An algorithm for adaptive background subtraction is then used to detect foreground objects. These detected objects are then tracked using a Kalman filter, in order to trace their trajectories and to detect the car events, “entering” or “exiting” a parking space.

**CHAPTER 2**

**LITERATURE SURVEY**

To solve the issue of parking in congested areas, various approaches and strategies have been suggested. Ming-Yee Chiu et al. suggested a method for calculating the number of available parking spaces by counting the vehicles at the checkpoint. The induction loop sensors are mounted underneath the road surface to perform the counting. While sensors were less expensive, were less affected by environmental conditions, and detected accurately, their installation was difficult and resulted in road damage. It was also difficult to keep it running in the event of a breakdown. Thirdly, the precise location of the free parking area cannot be calculated since the counting method does not provide accurate information; instead, it simply records the number of vehicles going through the checkpoints. Every parking space has these sensors installed underneath it. Wan-Joo Park et al. suggested that cars be equipped with ultrasonic sensors to scan for available parking spaces. The sensors in this method are easily influenced by weather conditions such as rain, temperature, snow, and a strong air breeze. Vamsee K. Boda et al. proposed another approach focused on wireless sensor nodes.This approach was less expensive, and it relied on wireless sensor nodes installed at key locations such as lane turns, parking lot entrances, and exits. The discrepancy between incoming and outgoing cars can be used to calculate the total number of cars in the parking lot. The entire parking area available for parking can be examined through a camera using vision-based methods, the data is then analyzed, and the result obtained will specify the exact number and location of available parking spaces. According to Zhang Bin et al., vision-based parking space detection methods are simple to set up, low in cost, and the detector can be easily modified. Furthermore, the information gleaned from photographs is extremely valuable. However, one of the vision method's flaws is that the accuracy is highly dependent on the camera's location. For parking space occupancy detection, Thomas Fabian suggested an unsupervised vision-based method.

**2.1 OBJECTIVES**

* The objective of the project is,
* To Detect Free Avail Slot for vehicle.
* To Detect avail space using CCTV.
* To display the avail free parking slot.
* To provide the exact parking details to the users.

**CHAPTER -3**

**PROBLEM DEFINITION**

* 1. **EXISTING SYSTEM**

In the modern society, there is an ever-increasing number of vehicles. This is leading to problems such as large urban parking lots becoming inefficient, increasing difficulty to find open spaces in busy parking lots, as well as the increasing need to devote larger areas of land for additional parking spaces

* 1. **DRAWBACKS**

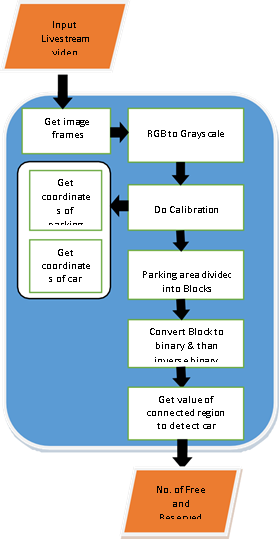
The drawbacks of the system are:

1. No provide parking Features.
2. Inefficient Systems.
3. No Intimation of the available parking slots.
   1. **PROPOSED SYSTEM**

* System will get a Livestream video of the parking lot from the camera.
* Images are captured when a car enters or leaves the parking lot.
* RGB Images are converted to grayscale images.
* Do calibration First select the coordinates of the parking lot. This will crop the extra space other than the parking lot from the image. Secondly, select the coordinates of the single parking slot. This will divide the parking lot into equal size slots.
* Each block is converted from grayscale to binary and then inverse binary to get the car in white color and parking area into black color.
* Threshold value is calculated in every block to detect whether that block contains a car or not.
* If a value is less than the threshold value then that block is free and available for parking cars and if a value is greater than a block is occupied.

**3.4 Advantages**

* Provides Correct information.
* Accurate in displaying Available parking slots.
* Easy to access.
* User friendly.



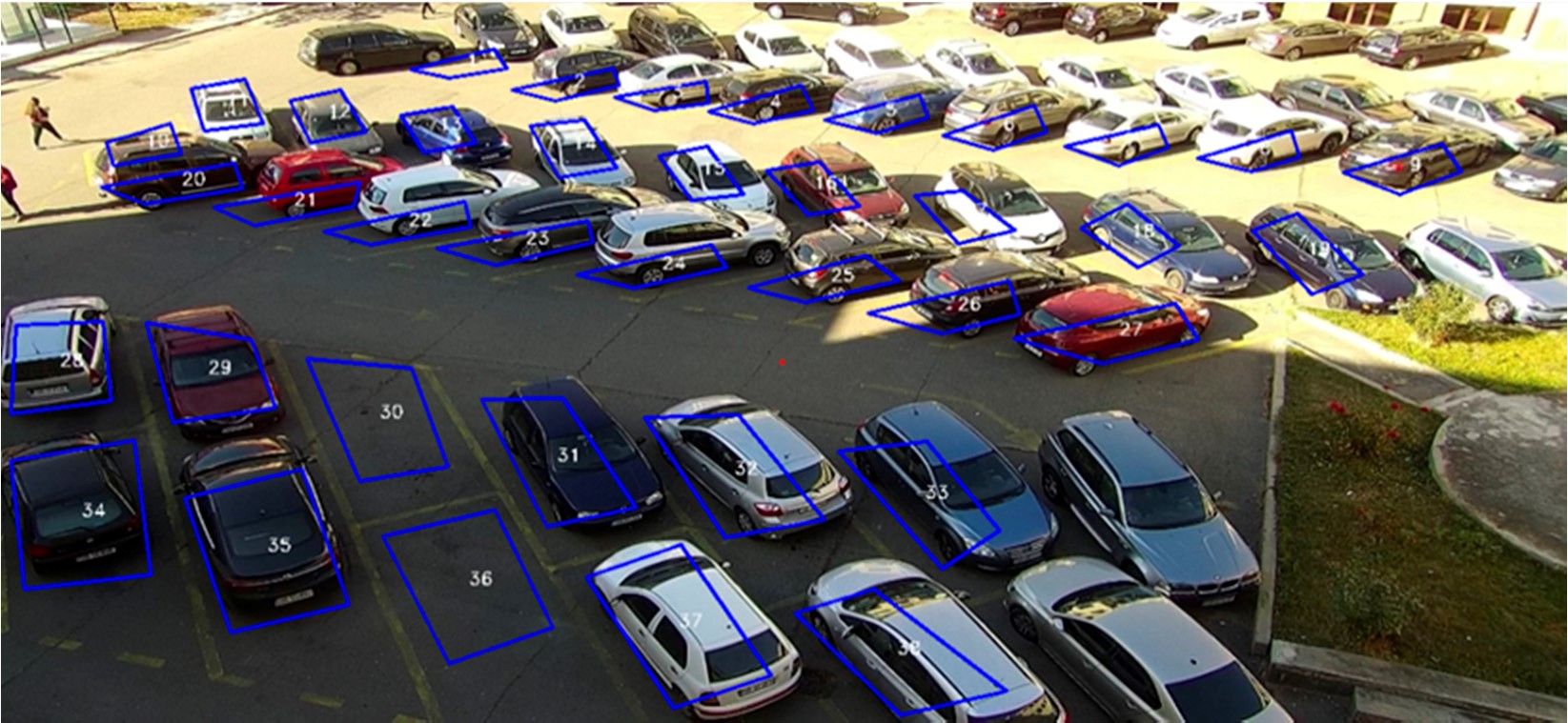
**CHAPTER 4**

**CONFIGURATIONS**

**4.1 INITIAL CONFIGURATIONS**

In order to determine the available spaces in a parking lot (how many spaces are available and their localization), it is necessary to obtain information concerning the structure of the monitored parking area. During this stage, a model for tracking the parking spaces on the pavement is provided. A graphic interface will be used for uploading an image with the observed parking area, and the user can trace the corners of each parking space. Using a pointer, four points for each parking space need to be marked. Once the provided points are saved, the system generates the map of the parking area which includes both the selected parking spaces, as well as their automatic numbering. This model is saved and will be reloaded for online use.





### 4.2. The Image Processing Chain

During this stage, the main goal of the proposed algorithm resides in assigning a status to each parking space based on the extraction of several different image features from the map provided in the initial configuration.

**4.3 Frame Preprocessing**

Preprocessing is performed on the images acquired from the video camera, in order to remove noise as well as to smooth the edges. During tests on both benchmarks and online processing, we have noticed the necessity of preprocessing to improve both efficiency and algorithm results. We apply SmoothMedian (size) and Smooth Blur (ksize) to the original image of the type RGB.

**4.4 Feature Tracking**

The standard deviation values for the three channels V (devSYUV), S (devSHSV), Cb (devSYCrCb) of the color spaces YUV, HSV, YcbCr, the number of corners (noSIFT), and the HOG descriptor mean (meanHOG) were used to create a history, based on predefined thresholds mentioned in Section 2.2.4.

This history tracks the availability of parking spaces.

Each value from metrics and measurements compares with a default threshold. Based on this comparison, the values of 1 (statusOccupied) and 0 (statusAvailable) are counted, and the status is determined by the predominant value.

Function SetStatus (indexParkingLot) if meanHOG[indexParkingLot] > 0.03 then statusOccupied++; else statusAvailable++; end if

if noSIFT[indexParkingLot] >= 7 then statusOccupied++; else statusAvailable++; end if

if devSYUV[indexParkingLot] > 1.4 then statusOccupied++; else statusAvailable++; end if

if (devSHSV[indexParkingLot] > 9) then statusOccupied++; else statusAvailable++; end if if (devSYCrCb[indexParkingLot] > 1.1) then statusOccupied++; else statusAvailable++; end if if (statusAvailable > statusAvailable) then status = 0

else status = 1 end if

EndStatus

For more accurate results, was have created a history of 20 frames (Figure 8), that memorizes the status of each parking space, obtained from measured metrics and measurement results. With each new frame, it is tested if the number of frames originally set has been reached. If yes, then adding the status to the list produces the effect of a “sliding window”, which requires the elimination of the first value of the buffer and the addition of the new value in the list, according to the FIFO principle. Thus, this technique allows for the last changes to each parking space to be retained in order to determine the status. Over time, this process helps stabilize changes in the background, such as the gradual change from day to night, different weather conditions.

for parkingSpace=0:totalNumberParkingLot if(sizeOfBuffer < 20)

Status = SetStatus(parkingSpace)

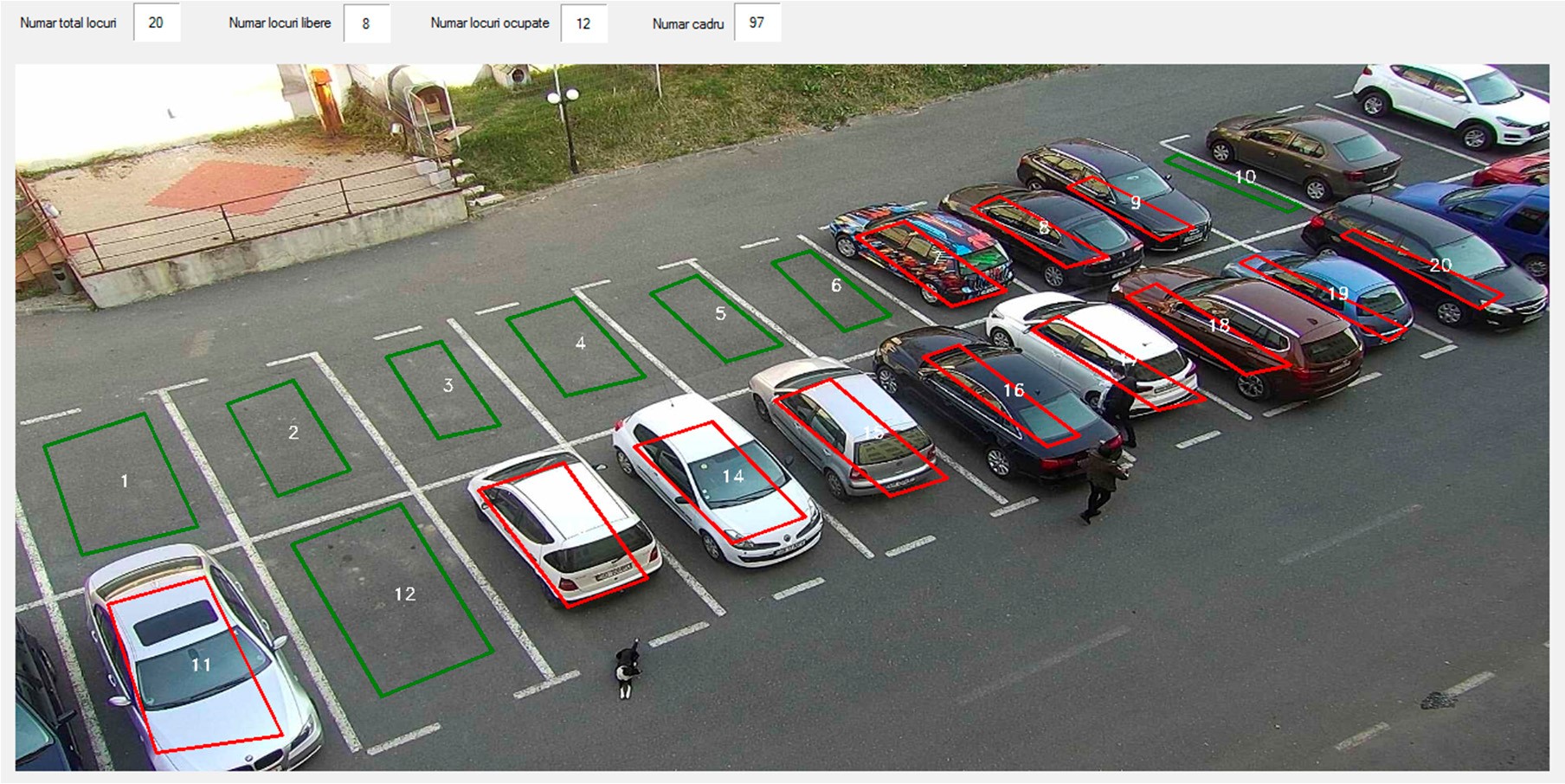
Add status in buffer else

SlidingWindow

Status = SetStatus(parkingSpace)

Add status in buffer end if if predominat is 1 in buffer StatusParkingSpace = 1 else StatusParkingSpace = 0 end if

end for



**CHAPTER 5**

**BENCHMARKS VALIDATION**

#### 5.1 Benchmarks and Metrics Used for Validation

The images used for testing purposes show two parking lots with 36 and 20 parking spaces, respectively, and were taken from fixed cameras. For this reason, the parking lot map is extracted only once, at system initialization. In view of an evaluation and without any benchmark available, we have created and manually annotated two of them, taken from the available cameras. Benchmark 1 has a total number of 90,984 frames, while the benchmark 2 was acquired with 121,400 frames, both at full HD resolution.The combination of the five methods used is the only one that has produced encouraging results in a good number of cases, but in the case of camouflage, cars overlapping the limits of a parking space, and under certain weather conditions (sun and shade), the assigned labels were wrong. System performance varies depending on meteorological conditions. However, accuracy is over 90%, making the system quite robust. The best performances are achieved in the absence of direct sunlight, with no shadows on the parking spaces.

For assessing the rate of parking spaces detection, the tests were performed in different moments, for each camera. One can find below the results from different benchmarks with the total number of frames.In order to evaluate the system performance, the following rates were calculated: false positive (FP), false negative (FN), true positive (TP), true negative (TN), and accuracy (ACC).

TP + TN

ACC =

FP + FN + TP + TN

where FP is the total number of available parking spaces labeled as occupied, FN is the total number for occupied parking spaced labeled as available, TP is the total number of available spaces correctly labeled, and TN the total number of occupied spaces correctly labeled.

**5.2 RESULTS AND DISCUSSION**

The proposed method is implemented by SPYDER. The online system gets images from the camera while the offline system gets images from the video file. The result of the online system shows that the proposed algorithms have effectively detected and notified the drivers of the available parking slots. The proposed algorithm is implemented on a model parking lot with space for 14 cars.In order to test the performance of our proposed algorithm, the accuracy of the system is measured with images taken at different time intervals. Performance is calculated by comparing the occupancy results to the ground truth every 5 sec. The performance of the proposed system is measured by the use of the equation (1)The accuracy of the proposed algorithm was found to be 100 percent, 98 percent 96 percent and 94 percent. The results show that when the captured images of the parking lot are not clear due to less lighting or occlusions, the efficiency decreases and the accuracy of the detection decreases. It is noted that the average performance is 99.5 percent and is very high compared to other applications for parking lot detection. The accuracy of the proposed work also depends on the type of camera used to monitor the parking lot. The correct detection of cars is shown in the blue color chart and the false detection of cars is shown in the orange colour.Cars in these virtual parking blocks and inform the drivers of the available parking blocks. The proposed work also depends on the type of camera used to monitor the parking lot. The correct detection of cars is shown in the blue colour chart and the false detection of cars is shown in the orange colour.Cars in these virtual parking blocks and inform the drivers of the available parking blocks.

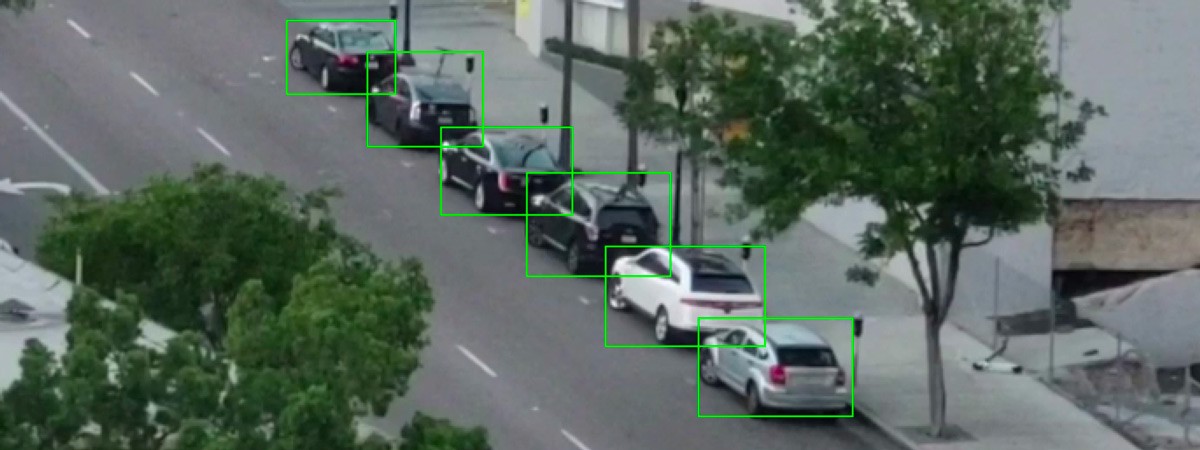
**TPS-**Total Parking Slots

**ANC-**Actual Number of Cars

**PNC-**Predicted Number of Cars

**Performance=**1-((|ANC-PNC|)/TPS)\*100 ---🡪1 eqn

**Percentage Error=** ((| ANC- PNC|)/TPS) \*100 ----🡪2 eqn



**Table 1:**

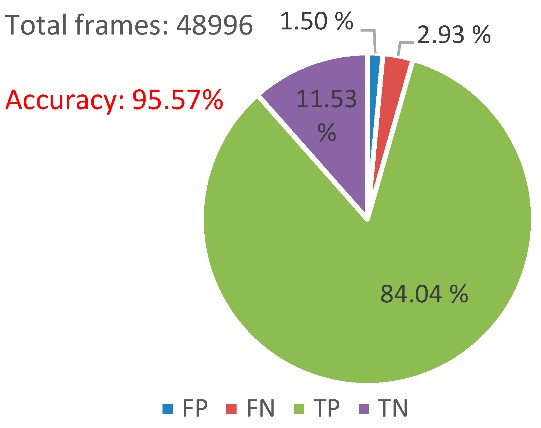
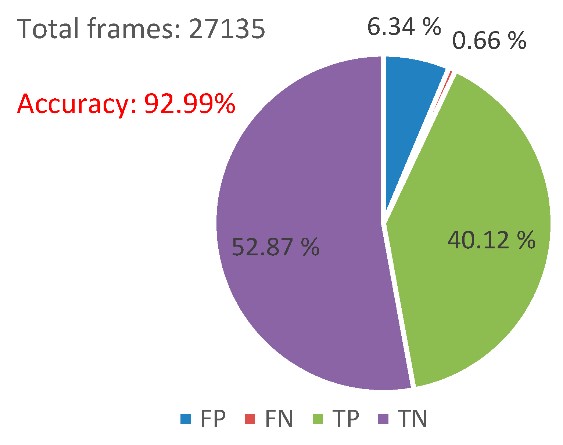
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Vehicle Appearance*** | ***Number of tests*** | ***Correct detection*** | ***False detection*** | ***Accuracy*** |
| **Sunny day** | Clear | 50 | 50 | 0 | 100% |
| Occluded | 50 | 47 | 2 | 95% |
| **Cloudy day** | Clear | 50 | 48 | 1 | 97% |
| Occluded | 50 | 46 | 4 | 94% |

The accuracy of the proposed algorithm is found to be 100%, 95% 97% and 94%.The results shows that when the captured images of the parking lot are not clear because of less lighting or occlusions, the efficiency decreases and the accuracy for detection’s reduces. It is observed that the average performance is 97.5 % and is very high as compared with other parking lot detections applications. The performance of the proposed algorithm in some cases drops down because of the strong shadows.

**Table 2:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Twin ROI***  ***detection method*** | ***Edge based***  ***detection method*** | ***Proposed approach*** |
| **Average detection Rate (car)** | 95% | 97% | 98% |

The accuracy of our proposed system is better than the Color based twin ROI detection technique, Edge based detection technique used in existing parking systems. In color based detection system and Twin ROI based system ,the efficiency goes down when the car and parking area is of the same color.. Our proposed method will automatically divide the parking area in multiple blocks by drawing virtual lines on the parking lot, efficiently detect the cars in these virtual parking blocks, and notify the drivers about the available parking blocks.



**CHAPTER 6**

**CONCLUSION AND RESULTS**

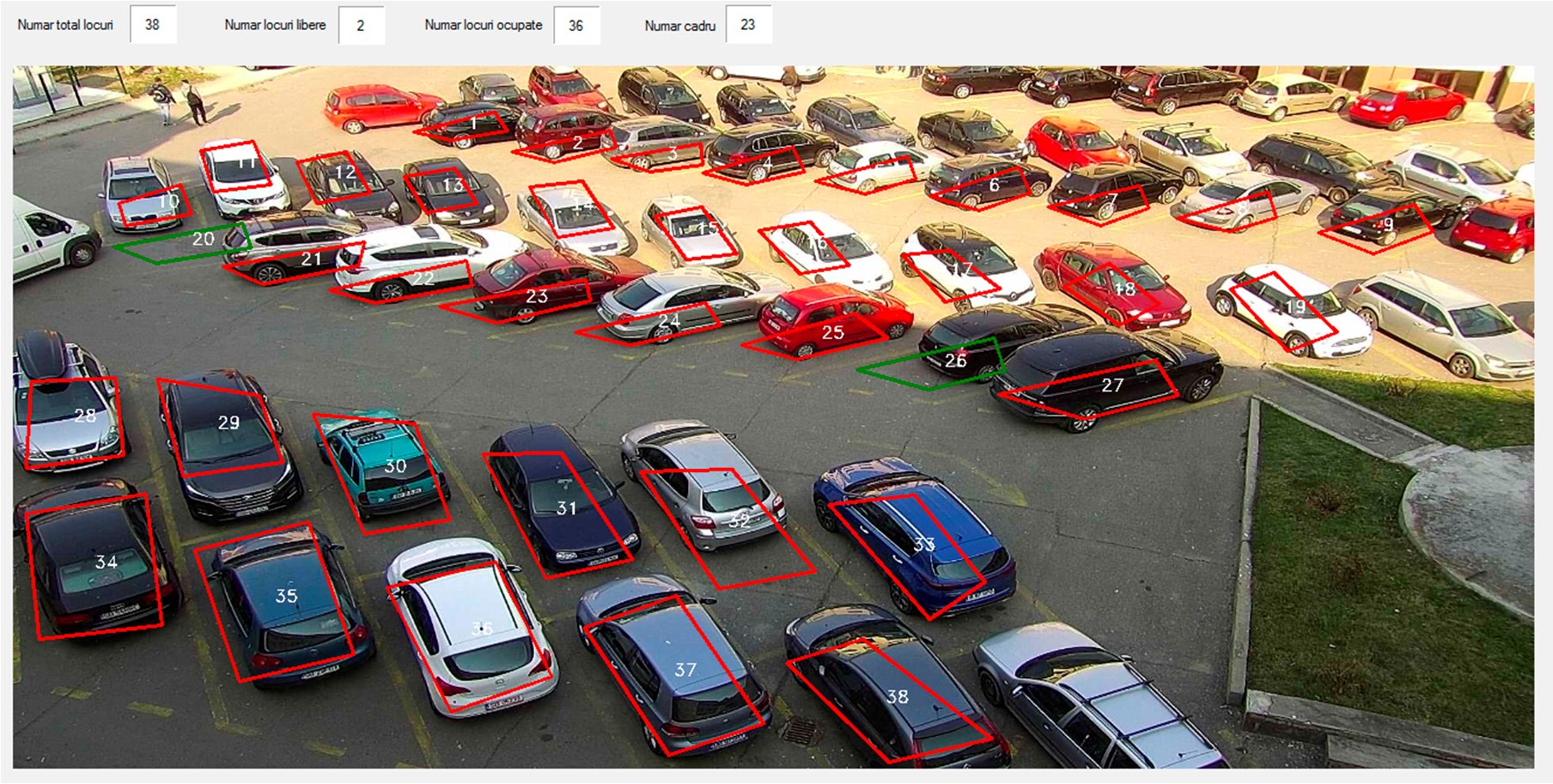
**6.1 CONCLUSION**

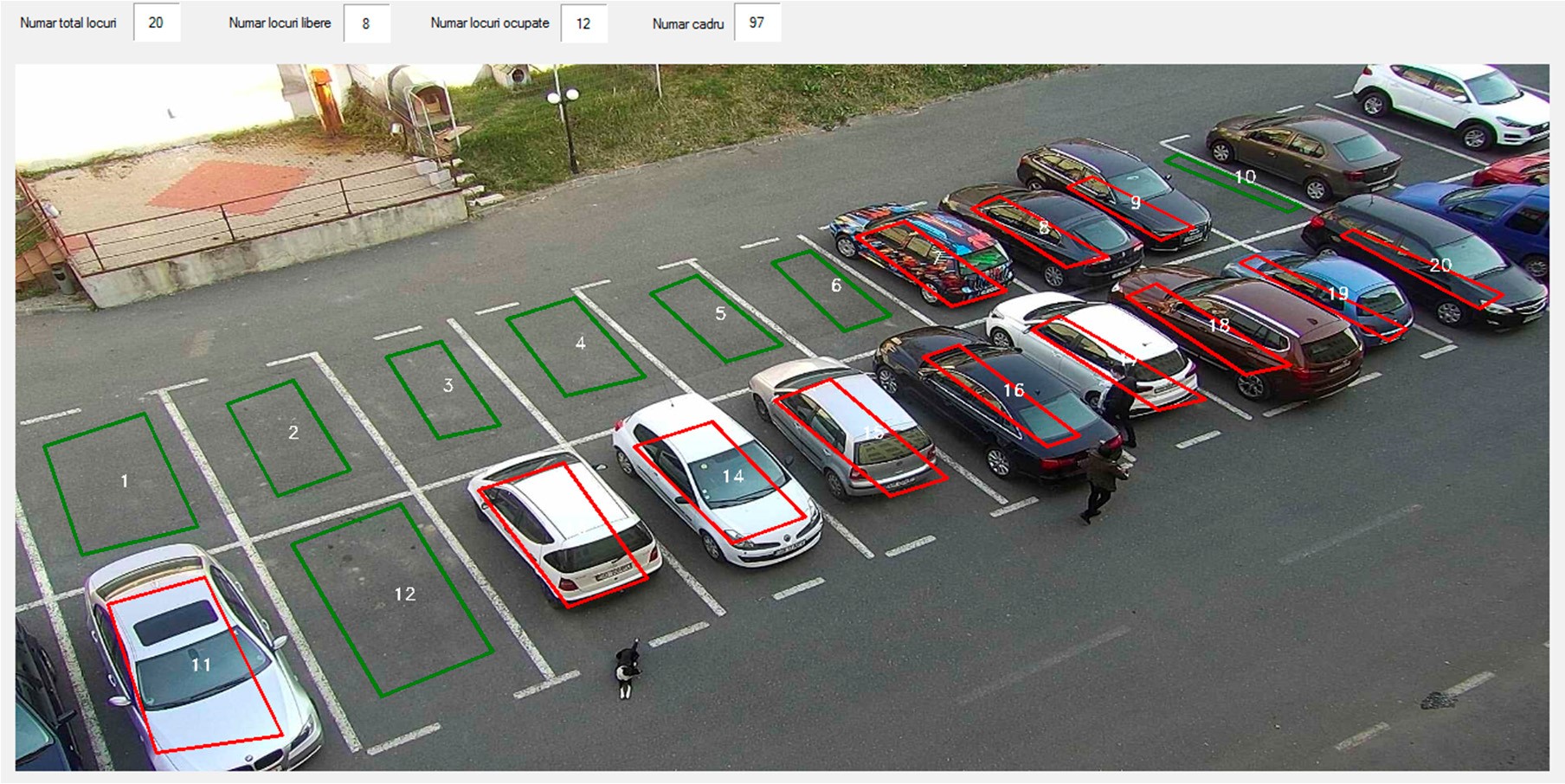
The fundamental commitment of this investigation is to advance the recognizable proof of accessible stopping openings to conceivably decrease the blockage in the stopping field. Because of headway in AI and vision base innovation practical programmed stopping frameworks encourage the drivers to find accessible spaces at stopping fields. Future specialists can zero in on portion explicit area to clients as of now enlisted from internet stopping the executive's framework. In this paper, a video detection system for parking lot occupancy has been described. Compared to other methods, the system uses a single camera sensor for the detection of multiple parking spaces occupancy. Results achieved using the two parking lot installed cameras, have an accuracy rate of over 93%. In order to exploit adaptive background, a mixture of Gaussian model was used. The background is processed using the HOG descriptor, the SIFT corner detector, and the HSV, YUV, YCbCr color spaces to solve various detection issues. These combined algorithms improve the accuracy of the parking lot occupancy detection. The applied algorithms have solved some of the background issues caused by the meteorological conditions, resulting in a high accuracy rate over a long period of time. The results with lower accuracy rate were due to unsolved issues, such as shade and camouflage, since the shade was identified as an object. However, over a long enough period of time, where there are intervals with different illumination conditions, the accuracy rate was over 93%.Due to the straightforwardness and robustness of the proposed system, a commercial implementation is foreseen. IoT technology and Smart City applications, the duality of the proposed method, as security surveillance and also feature extraction for parking space detection, have substantial industrial potential.

**6.2 APPENDICES**

**SCREENSHOTS**







**SOURCE CODE**

import numpy as np  
import cv2  
import yaml  
  
  
fn = r"\Users\user\Documents\DetectParking-develop\datasets\parkinglot\_1\_480p.mp4"  
  
fn\_yaml = r"\Users\user\Documents\DetectParking-develop\datasets\parking2.yml"  
fn\_out = r"\Users\user\Documents\DetectParking-develop\datasets\output.avi"  
config = {'save\_video': False,  
 'text\_overlay': True,  
 'parking\_overlay': True,  
 'parking\_id\_overlay': True,  
 'parking\_detection': True,  
 'min\_area\_motion\_contour': 60,  
 'park\_sec\_to\_wait': 3,  
 'start\_frame': 120}   
cap = cv2.VideoCapture(fn)  
video\_info = {'fps': cap.get(cv2.CAP\_PROP\_FPS),  
 'width': int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)),  
 'height': int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)),  
 'fourcc': cap.get(cv2.CAP\_PROP\_FOURCC),  
 'num\_of\_frames': int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))}  
cap.set(cv2.CAP\_PROP\_POS\_FRAMES, config['start\_frame'])   
if config['save\_video']:  
 fourcc = cv2.VideoWriter\_fourcc('D','I','V','X')  
 out = cv2.VideoWriter(fn\_out, -1, 25.0,   
 (video\_info['width'], video\_info['height']))  
with open(fn\_yaml, 'r') as stream:  
 parking\_data = yaml.safe\_load(stream)  
parking\_contours = []  
parking\_bounding\_rects = []  
parking\_mask = []  
for park in parking\_data:  
 points = np.array(park['points'])  
 rect = cv2.boundingRect(points)  
 points\_shifted = points.copy()  
 points\_shifted[:,0] = points[:,0] - rect[0]   
 points\_shifted[:,1] = points[:,1] - rect[1]  
 parking\_contours.append(points)  
 parking\_bounding\_rects.append(rect)  
 mask = cv2.drawContours(np.zeros((rect[3], rect[2]), dtype=np.uint8), [points\_shifted], contourIdx=-1,  
 color=255, thickness=-1, lineType=cv2.LINE\_8)  
 mask = mask==255  
 parking\_mask.append(mask)  
  
parking\_status = [False]\*len(parking\_data)  
parking\_buffer = [None]\*len(parking\_data)  
while(cap.isOpened()):   
 spot = 0  
 occupied = 0   
   
 video\_cur\_pos = cap.get(cv2.CAP\_PROP\_POS\_MSEC) / 1000.0   
 video\_cur\_frame = cap.get(cv2.CAP\_PROP\_POS\_FRAMES)   
 ret, frame = cap.read()   
 if ret == False:  
 print("Capture Error")  
 break  
   
   
 frame\_blur = cv2.GaussianBlur(frame.copy(), (5,5), 3)  
 frame\_gray = cv2.cvtColor(frame\_blur, cv2.COLOR\_BGR2GRAY)  
 frame\_out = frame.copy()  
   
  
 if config['parking\_detection']:   
 for ind, park in enumerate(parking\_data):  
 points = np.array(park['points'])  
 rect = parking\_bounding\_rects[ind]  
 roi\_gray = frame\_gray[rect[1]:(rect[1]+rect[3]), rect[0]:(rect[0]+rect[2])]   
   
  
 points[:,0] = points[:,0] - rect[0]   
 points[:,1] = points[:,1] - rect[1]  
   
 status = np.std(roi\_gray) < 22 and np.mean(roi\_gray) > 53  
   
 if status != parking\_status[ind] and parking\_buffer[ind]==None:  
 parking\_buffer[ind] = video\_cur\_pos  
   
 elif status != parking\_status[ind] and parking\_buffer[ind]!=None:  
 if video\_cur\_pos - parking\_buffer[ind] > config['park\_sec\_to\_wait']:  
 parking\_status[ind] = status  
 parking\_buffer[ind] = None  
   
 elif status == parking\_status[ind] and parking\_buffer[ind]!=None:  
   
 parking\_buffer[ind] = None   
   
   
 if config['parking\_overlay']:   
 for ind, park in enumerate(parking\_data):  
 points = np.array(park['points'])  
 if parking\_status[ind]:   
 color = (0,255,0)  
 spot = spot+1  
 else:   
 color = (0,0,255)  
 occupied = occupied+1  
 cv2.drawContours(frame\_out, [points], contourIdx=-1,  
 color=color, thickness=2, lineType=cv2.LINE\_8)   
 moments = cv2.moments(points)   
 centroid = (int(moments['m10']/moments['m00'])-3, int(moments['m01']/moments['m00'])+3)  
 cv2.putText(frame\_out, str(park['id']), (centroid[0]+1, centroid[1]+1), cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (255,255,255), 1, cv2.LINE\_AA)  
 cv2.putText(frame\_out, str(park['id']), (centroid[0]-1, centroid[1]-1), cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (255,255,255), 1, cv2.LINE\_AA)  
 cv2.putText(frame\_out, str(park['id']), (centroid[0]+1, centroid[1]-1), cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (255,255,255), 1, cv2.LINE\_AA)  
 cv2.putText(frame\_out, str(park['id']), (centroid[0]-1, centroid[1]+1), cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (255,255,255), 1, cv2.LINE\_AA)  
 cv2.putText(frame\_out, str(park['id']), centroid, cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (0,0,0), 1, cv2.LINE\_AA)  
   
  
   
 if config['text\_overlay']:  
 cv2.rectangle(frame\_out, (1, 5), (280, 90),(255,255,255), 85)   
 str\_on\_frame = "Frames: %d/%d" % (video\_cur\_frame, video\_info['num\_of\_frames'])  
 cv2.putText(frame\_out, str\_on\_frame, (5,30), cv2.FONT\_HERSHEY\_SCRIPT\_COMPLEX,  
 0.7, (0,128,255), 2, cv2.LINE\_AA)  
 str\_on\_frame = "Spot: %d Occupied: %d" % (spot, occupied)  
 cv2.putText(frame\_out, str\_on\_frame, (5,90), cv2.FONT\_HERSHEY\_SCRIPT\_COMPLEX,  
 0.7, (0,128,255), 2, cv2.LINE\_AA)  
  
   
   
 if config['save\_video']:  
 if video\_cur\_frame % 35 == 0:   
 out.write(frame\_out)   
   
   
 cv2.imshow('vechile lot detection by team ALPHA', frame\_out)  
 cv2.waitKey(40)  
   
 k = cv2.waitKey(1)  
 if k == ord('q'):  
 break  
 elif k == ord('c'):  
 cv2.imwrite('frame%d.jpg' % video\_cur\_frame, frame\_out)  
 elif k == ord('j'):  
 cap.set(cv2.CAP\_PROP\_POS\_FRAMES, video\_cur\_frame+1000)   
  
cap.release()  
if config['save\_video']: out.release()  
cv2.destroyAllWindows()

**6.3 REFERENCES**

1. Ming-Yee Chiu; Depommier, R.; Spindler, T.; , "An embedded real- time 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. T. Mar; N. Marcel;, “ Video-based parking space detection,” 2012 [Online]. Available: <http://www.ini.rub.de/data/documents/> tschentscherneuhausen\_parking\_space\_fbi2012.pdf
4. Ichihashi, H.; Notsu, A.; Honda, K.; Katada, T.; Fujiyoshi, M.; , "Vacant parking space detector for outdoor parking lot by using surveillance camera and FCM classifier," Fuzzy Systems, 2009. FUZZ-IEEE 2009.
5. Boda, V.K.; Nasipuri, A.; Howitt, I.; , "Design considerations for a wireless sensor network for locating parking spaces," SoutheastCon, 2007.
6. Fabian, T. , "An Algorithm for Parking Lot Occupation Detection," Computer Information Systems and Industrial Management Applications, 2008.
7. Ichihashi, H.; Katada, T.; Fujiyoshi, M.; Notsu, A.; Honda, K.; , "Improvement in the performance of camera based vehicle detector for parking lot," Fuzzy Systems (FUZZ), 2010.
8. Yusnita, R.; Fariza N. ; Norazwinawati B.; “Intelligent Parking Space Detection System Based on Image Processing,” International Journal of Innovation, Management and Technology, Vol. 3, No. 3, June 2012.
9. N. True;, “Vacant Parking Space Detection in Static Images,” Projects inVision & Learning, University of California, 2007 [Online]. Available: <http://www.cs.ucsd.edu/classes/wi07/cse190-a/reports>

/ntrue.pdf.

1. Najmi Hafizi Bin Zabawi, Sunardi, Kamarul Hawari Ghazali, “Parking lot detection using image processing method”, October 2013.