American Journal of Intelligent Systems 2017, 7(3): 77-80
DOI: 10.5923/j.ajis.20170703.08

SmartPark – Smart Parking Lot System

Bhavith Kumar, Shravan Kanagokar*, Trevor Loren, U. Akshay Kini, Vijetha U.

Department of Computer Science and Engineering, St Joseph Engineering College, Mangaluru, India

Abstract The fast pace of urbanization has led to increase in number of on-road vehicles. While most business and educational establishments have a parking facility, it requires an operator to be present at all times to verify the users accessing the parking lot. This paper discusses the implementation of a system which can be installed in parking lots to automate the process of verification of its users. The system uses image processing and character recognition to read license plates of users of the system and grant or deny access appropriately.

Keywords Image processing, OCR, Raspberry Pi

1. Introduction

With an increase in the number of on-road vehicles comes an increased demand for parking space. Present parking-lot systems in business and educational establishments require an operator to be present to monitor incoming traffic and verify users. SmartPark is a system which can be installed at such locations to automate the process of verification of users of the parking lot.

The system proposes to reduce labor and operational costs while also reducing delays and other inconveniences which may occur in a manual system.

The system runs on a Raspberry Pi card-sized computer which provides sufficient computational power to perform character recognition from images and also allows interfacing with various hardware components such as camera, distance sensor and liquid crystal displays. The easy availability of the components allows the system to be implemented at minimum cost.

2. Hardware Used

Cost reduction was a primary objective of this project. For this reason, easily available and low-cost systems were used for the implementation. Following are the hardware used to implement the system.

Raspberry Pi 3 Model B: A card-sized computer with enough computational power available on-board to perform the intensive calculations required for image processing. The GPIO pins allow interfacing with different hardware components [1].

Raspberry Pi Camera Module: Used to capture images of vehicles entering the parking lane.

Ultrasonic Sensor: An electronic device which uses sound waves to find the distance of vehicles in the parking lane.

Servo Motor: A motor which can be set to a particular position by varying the duty-cycle of electric pulses sent to it.

16x2 LCD Display: Used to display messages to the user [2].

3. Proposed System

The user of the parking lot has to first register their vehicle with the establishment. The user is then provided with a unique PIN which they must remember.

After registration the user simply has to park their vehicle at the parking lane. The system automatically captures an image of the license-plate of the vehicle and performs character recognition to obtain the license plate. This is then cross checked with the database to see if it is a valid user and the boom-barrier is opened appropriately.

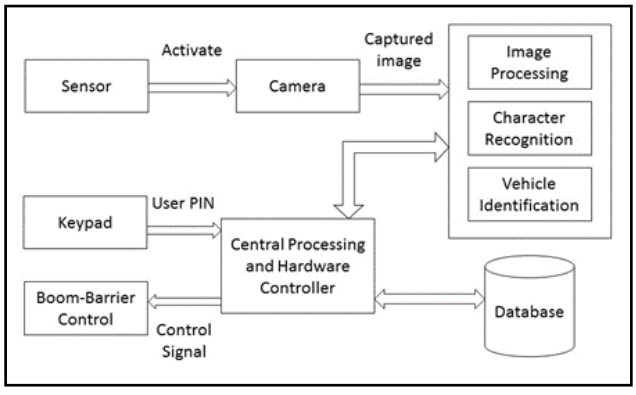

Figure 1. A sample line graph using colours which contrast well both on screen and on a black-and-white hard copy

Figure 1 shows a basic architectural diagram of the system. The sensor detects the presence of a vehicle in the parking

* Corresponding author:
shravan.sk95@gmail.com (Shravan Kanagokar)
Published online at http://journal.sapub.org/ajis
Copyright © 2017 Scientific & Academic Publishing. All Rights Reserved

lane. It initializes the camera and captures an image of the license plate. The image is then processed and characters are recognized. The result is checked with the database and the boom barrier is opened or kept closed appropriately. In case of failure in recognizing a registered license plate, the user can enter their PIN to verify their identity.

4. License Plate Recognition

OpenCV is the image processing library used to process images of vehicles entering the premises. License plates of the vehicles are isolated and then characters in the plate are recognized to produce the resultant license plate characters [3].

The proposed method to recognize and read license plates consists of 4 major parts: Processing the image and isolating the license plate, resizing the separated license plate image and processing it for character recognition, obtaining characters as separate images from the license plate and finally performing character recognition on the processed image to obtain the plate number.

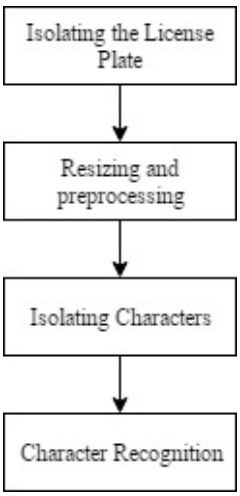

Figure 2. Steps in License Plate Recognition

4.1. Isolating the License Plate

For this system, the image will be obtained from a camera module. The image is then loaded to a computer readable format and sent for further processing.

The first step in processing the image is convert it from a RGB image to a greyscale image. This converts it to a more usable two dimensional array.

The second step involves removal of noise and smoothening the image using a bilateral filter. Filtering is perhaps the most fundamental operation of image processing and computer vision. In the broadest sense of the term "filtering", the value of the filtered image at a given location is a function of the values of the input image in a small neighborhood of the same location. A bilateral filter removes the high variation amongst adjacent pixels by assigning an average value to the pixel based on its neighboring pixel values but preserving edges at the same time.

Figure 3. Image captured for recognition

The third step involves applying contrast to the image by performing histogram equalization. Histogram equalization is used to stretch out the intensity range across underpopulated intensities. The image obtained from this is a contrasted image that has its high intensity peaks spread across a wider region.

The fourth step is to perform a morphological opening with a rectangular structuring element. This removes some of the foreground pixels from the edges of regions of foreground pixels but we preserve foreground regions that have a similar shape to a rectangular structuring element.

The fifth step involves obtaining the difference between the equalized image and the morphological opening image, which is an image which will have a set of defined rectangular objects with all the sharp non rectangular edges removed. A binary threshold is then applied to this image, the threshold value depends on the lighting conditions of the location of where the image was taken.

The next step is to mark all the edges and dilate them so that we get an image with a distinguished rectangle, which will probably be the license plate. The final step is to find the contour which has four edges with an aspect ratio that matches the aspect ratio of a license plate and crop that area from the main image. This cropped image will be the number plate in most cases.

Figure 4. Isolated license plate

4.2. Resizing and Pre-processing

The license plate image has to be resized so that size of every character feature on the plate can be isolated based on size relative to the image size. If the image size varies the character size may vary too. The aspect ratio of the plate should also be maintained so that the character features may not get warped during the resizing phase. To accomplish this, a target pixel area is set for the plate such that character features are distinguishable from each other. The image is resized within the set pixel area adjusting the width and height to match the aspect ratio of the original license plate image. Reducing image size also reduces the execution time for further processing steps and for character recognition.

Further processing the image is essential to distinguish the characters on the license plate from the other features on the license plate such as- screws, blemishes, country specific logos, and so on. The greyscale version of the resized license plate image is used.

First, a Gaussian blur to smoothen out the image, followed by applying contrast to the image through histogram equalization. Next the colors are reduced in the image by splitting it into RGB channels and clustering the image and applying a centroid value of these channels to all pixels so that the resulting image will have a specified number of colors. The third step involves applying a threshold to the image. The threshold values may vary based on glare and lighting conditions, hence the plate image must be clicked under ambient and constant lighting conditions. Reflective license plates provide better results when there is a light source attached to the camera module. Finally, an eroding function is applied to get a crisper set of character features on the license plate image after applying threshold.

4.3. Isolating Characters

To implement character recognition it is desirable to separate the characters on the license plate as separate images. The plate image obtained from preprocessing in the previous section is used as an input.

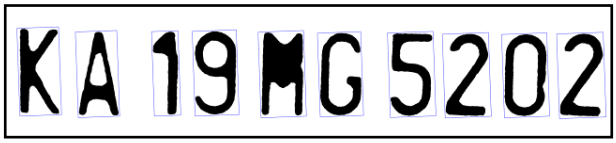

Figure 5. Isolation of characters on the plate

The image is inverted using a bitwise NOT function. The next step is to find all the contours in the image and separate every contour within a given size range which corresponds to the size of a prospective character on the plate. A list of the contours forms the list of characters on the plate. To further clean up the license plate and leave on the characters on it, a mask image excluding all the area on the plate except the characters is made and a bitwise AND function is applied with the original image to obtain a plate image with only character features.

4.4. Character Recognition

This involves comparing the character images obtained from the previous step with character data that is present in a data file. The character images are matched with a template that most resembles it amongst a set of similar templates. Every template has an associated classification letter or digit. These letters and digit form the license plate number that is required by the system.

The data files containing character information can be made using the same functions used in recognition but, an additional step must be added in place of optical character recognition to generate and classify character based on user input.

5. Database

The obtained plate after the plate recognition can be stored in database for future reference. SQLite is used to implement the database in this system. SQLite is a standalone database and does not follow client-server model. Thus, it is suitable for small standalone applications. It is also cross-platform and ACID compliant. SQLite is also available for most of the high-level programming languages.

The license-plate number can be stored in the database either manually or during plate recognition and can be compared with the database to know which plates belongs to whom. This provides an additional facility to existing plate recognition and may be helpful to scale the system to include paid-parking functionality and theft-detection.

6. Controller Process and GUI

The controller process runs on the Raspberry Pi and controls the functioning of the entire system. The GUI program can be opened on any computer connected to the same network. The GUI includes options to register and de-register users, bypass the system in case of emergencies and manage administer accounts.

The controller program listens to a TCP port for requests from the GUI. It also controls the complete functioning of the system during normal operation.

The controller polls the distance sensor for the presence of a vehicle in the parking lane. If a vehicle is present it clicks an image using the camera module and then runs the image processing module. The result from image-processing is the license plate number which is checked with the database and the boom-barrier is opened.

When registration process is called from the GUI, the normal workflow is interrupted to register the vehicle. The license plate is read and can be registered into the system.

In case of emergencies, the administrator can bypass the entire system and make way for ambulances, fire trucks, etc. if required.

7. Implementation and Results

The system is implemented using the Raspberry Pi 3, a 5 mega-pixel camera module board to capture vehicle images and an ultrasonic sensor to detect a vehicle and trigger the license plate recognition process. The license plate recognition process uses OpenCV libraries for processing and extracting character data from the license plate image and the implementation software is completely coded using python 2.7. The standalone GUI for system administrators is implemented using Jython. A SQLite3 database maintained on the Raspberry Pi is used as a backend database to store user data and vehicle logs. A 16x2 LCD display is used for user interaction during the vehicle entry process. A servo motor is used to simulate the presence of an entry mechanism

that would have been represented by a boom barrier in real world parking lots. The system on implementation achieved successful results and it passes all software and hardware tests that were put together to meet the requirements.

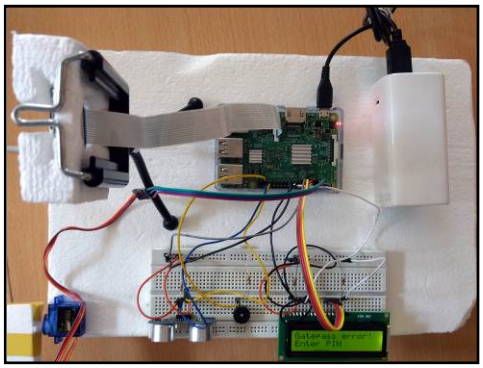

Figure 6. Isolation of characters on the plate

Table 1 shows the results achieved using sample government issue Indian license plates and license plates that follow similar formats with slight variation in fonts and sizes in a well-lit environment.

Table 1. Observed Results from Implementation

Success Rate for license-plate recognition	~85%
Execution time	~8 seconds
Maximum distance of vehicle for current setup	60 centimetres

8. Conclusions

The system proposes to replace current manual record keeping with an automated system which keeps tracks of vehicles entering the premises, thus reducing labor costs, delays and inconveniences to the users. The prototype model successfully achieves all the proposed objectives and is able to deliver a consistent and reliable service with minimal operating costs. The system can be extended to include keeping track of vehicles exiting the parking lot. Additional functionality such as theft-detection and paid-parking can be implemented. Improved algorithms and image processing techniques can be devised which will improve the quality of service.

REFERENCES

[1] Raspberry Pi User Site Available: https://www.raspberrypi.org.

[2] Riley, M., 2012. Programming your home: automate with Arduino, Android, and your computer. Pragmatic Bookshelf.

[3] Bradski, G. and Kaehler, A., 2008. Learning OpenCV: Computer vision with the OpenCV library. "O'ReillyMedia,Inc."