Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 6260372, 7 pages
http://dx.doi.org/10.1155/2016/6260372

Research Article

gPark: Vehicle Parking Management System Using Smart Glass

Rana E. Ahmed

Department of Computer Science and Engineering, American University of Sharjah, Sharjah, UAE

Correspondence should be addressed to Rana E. Ahmed; rahmed@aus.edu

Received 19 December 2015; Revised 23 February 2016; Accepted 3 March 2016

Academic Editor: Fanli Meng

Recent advances in wearable technologies have opened new avenues for their applications in various fields. This paper presents the design, implementation, and testing results for a vehicle parking management system using smart Glass technology. The management system consists of four major interconnected applications. The most important one, running on the smart Glass, scans the vehicle number plate and extracts the related information in real time. The vehicle information is sent to the remote server for checking of any violation. The server sends the updates back to the Glass that allows the parking attendant to take further actions, if needed. The system was tested in real-life scenarios, and it was found that the detection accuracy up to 75% can be easily achieved with current hardware and software capabilities of the Google Glass.

1. Introduction

Traditional vehicle parking management systems are becoming inefficient and require a lot of human interaction. When one wants to park a vehicle in a paid parking lot, he/she has to either purchase a ticket for a specific amount of time or in some cases has a parking sticker allocated for certain time duration (e.g., day/month/year). In both the scenarios, the parking attendant has to walk up to the vehicle's windscreen and check for the displayed ticket or sticker. In case of any violation of rules, the parking attendant notes down the vehicle's license plate and issues a fine and places a fine slip on the vehicle's windscreen.

Recent advancements in wearable technologies can help us design better and efficient parking management system. This paper describes a novel solution to the above-mentioned problem using smart Glass, more specifically, the Google Glass. In our solution, named gPark, the parking attendant does not need to walk up to a vehicle's windscreen; rather, the attendant just scans the vehicle's license plate using Google Glass. The application running on the Glass processes the vehicle license plate and extracts the vehicle details in real time. The vehicle details are then sent to the server to check for any parking violation. The information displayed on the Glass screen could be any violations registered to the system, which helps not only the parking attendant but also the law enforcement agencies, if needed.

Google Glass is the latest wearable technology in the market today. In January 2015, Google announced that it would stop producing the prototype but remained committed to the development of the Glass. The Google Glass has a built-in camera, in addition to other two input methods. One input method is by using the capacitive touch pad placed on the right side of the Glass. The change in capacitance allows the touch pad to respond to the user's instructions. When a user's finger comes in contact with the touch panel, a controller chip placed internally detects the variation in capacitance and registers it as a touch. By swiping the finger horizontally, it allows the user to navigate through the menu list on the device. The other method of input is by giving in voice commands. Google started deploying custom developed applications for the Glass, called Glassware. The two development techniques available are Mirror API and Glass Development Kit (GDK) [1–3].

The Mirror API allows the interaction of the Google Glass with a web-based service, Google's cloud API and the developed web-based services talk to each other, and then communication with the Glass itself takes place. The GDK is Android-based and does not require a network to run on the Mirror API. Moreover, apps developed using GDK run endemically on the Glass rather than on the server.

To the best of our knowledge, there is no published work available in the public domain on the design of the parking management system using Google Glass. The following are some related works reported in the literature.

In [4], a neural network-based artificial vision system, Visicar, is proposed. This system is able to analyze the image of a car given by any camera to locate the registration plate and recognize the registration number of a car. The system analyzes the image with the purpose of finding and recognizing the car number plate.

Several license plate recognition methods have been proposed in the literature [3, 5–8]. Those methods use still images or videos to process the number plate information. The most widely used system till now is the one in which the parking officials carry handheld Personal Digital Assistants (PDAs) to record the details of the vehicle in the case of any violation. The PDA is connected to a server where these details are sent and a fine is recorded against the owner of the particular vehicle, in case of a violation [9].

The remainder of the paper is organized as follows. Section 2 provides details about the system architecture of our application with details about its hardware, software, and networking components. Section 3 describes the working of one important component of the application, the Google Glass Application. We provide test results and discussions on the results in Section 4, followed by Conclusions in Section 5.

2. System Architecture

2.1. gPark Application Overview. The parking solution, gPark, described in this paper involves multiple portable devices such as the Google Glass and mobile phone. The gPark Application is spilt into four major components:

(i) gPark Google Glass Application.

(ii) gPark Web Service/Management.

(iii) gPark Mobile Communication.

(iv) gPark Cloud Sync.

The components are described briefly next.

2.1.1. gPark Google Glass Application. The Google Glass is loaded with an automatic number plate recognition (ANPR) processing system, and it uses the built-in camera on the Glass to capture a live feed and process it for any presence of license plates. The primary purpose of the Google Glass Application is to develop a number plate identification system, which identifies the vehicle's number plate. The parking attendant can wear the Google Glass and can scan vehicles in real time. Once the camera identifies the car's license plate and captures it, the application extracts the license plate details (e.g., F 7193), converts it into raw digital format, and pushes it to the backend database via HTTP POST and GET methods for further processing.

2.1.2. gPark Web Service/Management. The gPark Google Glass Application communicates with the web interface requesting for information of vehicles registered in the system. The volumes of requests that are handled by this system can be massive and time dependent (e.g., heavy requests during certain times of the day). The backend database contains the users' records and the registered vehicles. As the algorithm queries the database, trying to match the data to the acquired number plate, the user's name, status, and license plate number are pushed back to the Glass, once a match is found. The push methods TO and FROM the Glass are implemented using a cloud-based server using APIs to send and receive the data. In addition, the application allows the administrator to manage the vehicles and users and administer the parking facility. This application is built with PHP and HTML front end and MySQL DBMS as the backend. The web service is deployed on Microsoft Azure cloud platform, which can be scaled up/down based on client demand. Furthermore, once the Google Glass reads vehicle's details, it tags the GPS coordinates (latitude and longitude) to the server. This helps record location-based history of any vehicle that is tagged by the system.

2.1.3. gPark Mobile Communication. The functionalities available on the Google Glass (ANPR algorithm) and the web interface are replicated in the mobile application, making it a cross-platform application. In cases where the Google Glass is not available, the same process can be carried out using the mobile phone. In addition, the application also allows parking attendants to issue a fine on a vehicle that has any registered parking violations.

2.1.4. gPark Cloud Sync. gPark Cloud is an integrated component that allows gPark to coordinate and seamlessly synchronize data among different components mentioned above. It coordinates the movement of data among these components by providing a centralized cloud platform. gPark Web, gPark Glass, and gPark Mobile companion interface with the gPark Cloud to ensure that the same data is accessible across all devices. For instance, gPark Glass interfaces with gPark Cloud to update the information captured from a license plate. The same information is accessible on gPark Web as well as gPark Mobile. A Web Service, running on Microsoft Azure, is designed using Java Tomcat and facilitates this feature. The database and the application logic reside behind this cloud platform.

2.2. Technical Approaches. There are two approaches to achieve image processing related with plate number recognition:

(i) *Remote Image Processing*: sending the photograph of a license plate to the backend server and then running an image processing algorithm at the server, or

(ii) *Local Image Processing*: running the OpenCV (Open Source Computer Vision) image recognition engine [10] on the Google Glass processor.

2.2.1. Remote Image Processing. One of the major constraints during the formulation of this design approach is the limited processing power of the Google Glass. Additionally, higher consumption of battery power during image processing in the Glass is a major setback. In order to solve this particular

Table 1: Network data rate comparison.

Network	Data rate (KBps)	Transfer time (500 KB)	Transfer time (5 KB)
Etisalat EDGE	10.5	~48 seconds	~0.5 seconds
DU EDGE	11.1	~45 seconds	~0.5 seconds

issue, the *Remote Image Processing* approach suggests that the photograph (.jpg file) of the car plate is sent to a remote web server, where battery life and processing power are not a constraint and the image recognition algorithm is run at the server side. This would help to solve the problem of draining battery power and reduce the complexity of running compute-intensive image processing applications on Google Glass. The following are some advantages using this approach:

(i) There is longer battery life for Google Glass.

(ii) Complexity and accuracy of image recognition can be better since the image recognition component of the system resides on a powerful server where CPU speed and memory size are not limited.

The major disadvantage of the approach is that sending .jpg images from the Glass to the web server requires high network bandwidth and reliable Internet connectivity. This can be challenging in the areas where there is poor or limited connectivity to the Internet.

2.2.2. Local Image Processing within Glass. OpenCV (Open Source Computer Vision) [10] is a library of open source, optimized, programming functions aiming at real time computer vision. The application areas of OpenCV for our research include 2D and 3D feature estimation and recognition, segmentation, and HCI (Human Computer Interaction). In order to support above-mentioned application areas, the OpenCV includes statistical machine learning algorithms library. The algorithms include support vector machine (SVM) and artificial neural networks (ANNs).

The Local Image Processing approach is to develop the gPark Application in the Glass that also includes OCR and OpenCV integration. This approach would help process and read characters from images within the Glass itself. The data that would be sent to the server would be in plain computer readable text (instead of image in .jpg file, as in the remote processing method). Major advantage of this approach is that the information sent to the server is in plain text with 10 characters or less, which does not require high-speed network connectivity. The approach is ideal for most situations as connectivity is via a mobile device. Of course, major disadvantage of the approach is that the image has to be processed within the Glass hardware/software system. This is complex and challenging as image processing on Google Glass is limited by memory, operating system openness, and CPU processing power.

After thorough research and considering real-world scenarios, we decided that the *Local Image Processing* in Glass approach was ideal for our situation. One of the major reasons for this was the Internet connectivity limitation on the move.

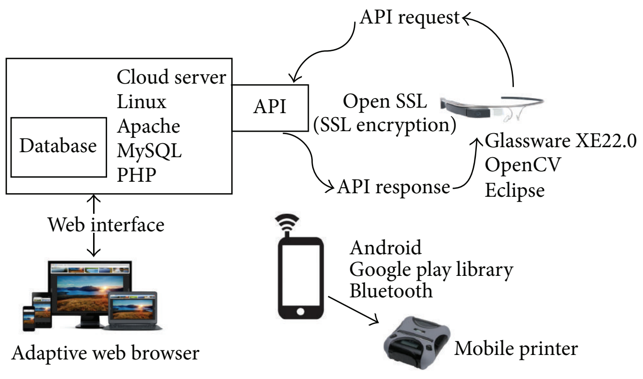

Figure 1: Overall system architecture.

In areas where the network data availability is limited, transfer rates of information can be extremely slow, making the usage of Google Glass for license plate detection impossible. We conducted speed tests on EDGE networks provided by telecom networks in the United Arab Emirates. Table 1 shows the results of the speed test to transfer images of sizes 500 KBytes and 5 KBytes.

2.3. Overall System Architecture. This subsection describes the use of various hardware and software components, which contribute to the development of the gPark Application. Figure 1 shows overall system architecture of gPark.

2.3.1. Hardware. The hardware components in the system are Google Glass, a Linux based backend server running on the cloud and a mobile device that is paired with the Google Glass via Bluetooth for Internet connectivity, and a portable printer to the mobile device for printing fines tickets.

The processor present in the Google Glass is OMAP 4430 System On Chip, Dual Core Processor. The Glass has 5 M pixels' camera. One way for a user to communicate with the Glass is by voice commands. The user can speak into the built-in microphone on the Glass, and the speech is then recognized and interpreted by the microprocessor. For example, the "OK Glass" command, which activates the Glass, is within the *Image/fs/data/com.google.glass.voice* directory on the Glass itself. This directory consists of all the information about the hands-free voice commands. There are many options the user can choose from, such as taking a picture, recording a video, Google searching, messaging, making a phone call, or asking for directions [11]. In the case of the gPark system, the user needs to activate the application by giving a voice command "Start gPark" or can use the touch pad to select the gPark Application on the menu screen.

The Glass is paired with a mobile phone via Bluetooth for basic configuration and setup. This is necessary as the Glass lacks a keyboard input method. The Glass processes the live feed of the camera to identify license plates and the data is sent to the cloud server over HTTP.

We used Ubuntu 14.04 LTS based server running as a private cloud instance, along with Apache, PHP, and MySQL that store information about all users. Upon API request by the Glass, the server sends or receives specific information.

A mobile device (Android OS preferred) is paired with a portable Bluetooth printer. The mobile device connects over the Internet to the server to access the backend database through gPark Cloud and receives requests for printing any fines tickets. It also includes administrator interface where the admin can manage users and cars registered in the system.

A portable printer, Wireless Monochrome Printer, is paired with the mobile device over Bluetooth. The application sends print commands at the request of the user.

The software part of the system primarily is composed of the following four major components (applications) that are integrated together.

2.3.2. gPark Glass Application. Using the built-in camera of the Google Glass, the vehicle is scanned in real time. In order to extract the number plate, ANPR algorithm is developed on the Glass in Java, using the following software tools:

(i) Android GDK (Glass Development Kit): This is an add-on feature to the Android SDK that allows developers to build Glassware that runs natively on the Glass [12].

(ii) OpenCV Engine 2.6 for image processing: OpenCV is a library or a collection of software algorithms used for both academic and commercial purposes for computer vision applications. OpenCV is released under BSD license. OpenCV mainly focuses on real time applications and demonstrates how to accelerate these logical algorithms on hardware. OpenCV supports several interfaces although C++ is the primary language. Some of the other interfaces are C, Python, and Java, and it supports Windows, Linux, Mac OS, iOS, and Android. The library consists of more than 2500 optimized algorithms, which can be used to detect images and faces, categorize human movements in videos, follow eye movement, recognize scenery, and so forth [10].

(iii) Optical Character Recognition (OCR): OCR is the process of converting a scanned image into machine-readable/editable format. The main objective is to simulate human reading skills so that the computer can read, understand, and edit in a similar manner.

The Glass Application is a very critical application in our system. More details about its working are presented in Section 3.

2.3.3. gPark Web Application and Cloud Sync. The web application has two components: the backend database and the front end (website). In order to host the website, a Virtual machine (VM) is used to set up the cloud server. The software tools required are as follows:

(i) Ubuntu 14.04 LTS: operating system that will be running on the server.

(ii) XEN VPS: This is a virtualized server and is used for hosting a website online; managing and setting up of a server can be difficult and expensive; on the other hand, in VPS, a space is provided on the web server and all that is needed is to upload the files to this server; VPS hosting is similar to Virtual machines, where several operating systems run on one machine.

(iii) Apache web server.

(iv) OpenSSL for secure HTTPS access.

(v) PHP 5.0: PHP is a scripting language on the server side.

(vi) HTML 5.0 elements: Hypertext Markup Language, HTML, is language used to create web pages.

(vii) MySQL: In order to create the backend database, MySQL software is used; creating tables, inserting records, and performing search results or any other query is implemented using the SQL language.

(viii) Microsoft Azure: It is an open, flexible cloud-computing platform consisting of a collection of services such as computing, networking, and storage; it helps to make the system faster, efficient, and cost effective; Azure provides Infrastructure-as-a-Service (IaaS) and platform-as-a-service (PaaS) allowing developing, deploying, and managing application in any way; it also provides a scalable platform and supports several operating systems and programming languages.

(ix) Apache Tomcat: It is an open source web server and servlet developed by the Apache; Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and delivers a "pure Java" HTTP web server ecosystem for Java code to execute.

2.3.4. gPark Mobile Application. The software tools used to implement the application on the mobile phone are as follows:

(i) Android SDK: In order to develop on the Android platform, developers need the Android Software Development Kit (SDK); the SDK includes a variety of development tools, an emulator, and necessary libraries for development purposes.

(ii) Google Maps API: In order to embed the functionality and services of the Google Maps into the website and mobile applications, Google Maps API is needed.

(iii) Google Play Libraries.

2.4. Image Processing Workflow. The gPark Application running on Google Glass will try to localize the number plate; that is, it will try to find and isolate the number plate by surrounding it with a green box. Once a number plate is isolated, the plate is adjusted for its brightness and contrast. In addition, the tilt of the plate is taken into consideration and the dimensions are adjusted. Next step is the character separation, where it tries to locate the individual characters on the number plate and then runs OCR to detect these characters. Since ANPR systems are country-specific, a final analysis is required to check for the characters and their positions depending on the country-specific standards. In order to make it accurate, an average of the recognized values is taken over several images.

2.5. gPark Cloud Sync (Web Service) Architecture. The system has a decoupled architecture where the individual components of the system are separated in a modular fashion. Multiple instances are hosted on application servers, where different cache, database, and file servers support it. This architecture is replicated across multiple availability zones for better Quality of Service (QoS). Additionally, elastic load balancing ensures that the traffic from end user is balanced effectively. Moreover, CloudFlare for DNS will enhance security of the front end access. On the backend, the billing and the app deployment control the instances that are deployed on client demand.

2.5.1. gPark Client Manager. Superuser administration interface is designed for the product owners as well as the clients who would like to purchase gPark for their organizations. The client management interface contains both administrative features of the services clients have purchased from gPark and accounting and billing based on usage (number of users/cars registered in the system) as well as overall application control such as the ability to edit global settings including their web domain selection and organizational settings. This is backed with a database containing all the client information as well as their billing information for management purposes.

2.5.2. gPark Web Application. This service, provided by gPark to its clients, is an instance of the web application that contains all the functionalities such as vehicles, users, and violation management. It also includes a data input/output interface that lets clients use external third-party devices such as portable devices, mobile phones, and wearable devices such as Google Glass into the system for data retrieval and input. These are individual instances that are mirrored for every client who signs up and requests the gPark service.

3. The Google Glass Application

The Glass Application is the most important component in the gPark system, as it captures plate images and processes it. In order to implement the automatic number plate recognition (ANPR) system on the Glass, two techniques are required:

(i) Capturing the image in real time and locating the number plate and using OpenCV to process the raw images.

(ii) Segmenting the characters in the region of the license plate.

(iii) Optical Character Recognition (OCR) to recognize characters on number plates.

3.1. Plate Localization. Locating the license plate is one of the primary requirements of the ANPR Algorithm, since it gives a visual cue to the Glass user that a plate has appeared in the field of vision. This is also critical in the rest of the steps that follow in the ANPR process. The quality and accuracy of the system primarily depend on how efficiently and effectively the application detects the location of a license plate once it has appeared in the frame. The following are three major industry standard approaches to implement this process [13]:

(i) *Pattern matching method*: this method considers the entire license plate to be an object that has a separate frame on its own. Once a particular pattern is matched, it can be passed to a trained set of data that is similar to the pattern initially matched.

(ii) *Morphological method*: this method allows the detection and extraction of a license plate based on certain characteristics, such as its symmetry, brightness of the region, and colors.

(iii) *Hough transformation method*: this is one of the most common methods where a potential license plate region is found based on the characteristics of the edge-detecting pairs of straight lines that are assembled parallel.

The gPark Application uses a combination of the three above-mentioned methods to have a relatively low-power consuming plate detection method, which a low powered device such as the Google Glass can handle easily.

gPark uses standard OpenCV functions that are available for Android OS for the image processing. The application uses open source frameworks that allow OpenCV functions to be accessed in a Java environment.

Once the application is initiated, it executes in the active scanning mode, which fetches the images from the live feed of the camera and passes it to the Plate Detection Mode. The application then carries out the identification process and the recognized characters are then sent to the web server along with the current GPS coordinates using HTTP POST/GET methods, in order to retrieve the user information of a specific license plate.

gPark, unlike other ANPR based applications, relies on real time video feed that is available from the device camera. The feed, on a frame-by-frame basis, is transferred instantly to the PlateView function, which recognizes the image as a byte array.

The image, which is received as a byte array from the active scanning mode, is processed to find the area that contains the number plate using a combination of the plate localization methods mentioned above. Once a potential area of the license plate is detected, the OpenCV draw method draws a rectangle (green color) around the region of the number plate to alert the user that the system has found a potential license plate in the view. During the period the application is active, all license plates that are detected are stored as byte arrays. If the detected image has not been processed before by the Optical Character Recognition (OCR) function, the image is sent for OCR.

OCR is the most important component in the system, which is responsible for recognizing the characters in the license plate and converting them into machine-readable format. The following steps are carried out to achieve the desired recognition process:

(1) Converting the image into gray scale.

(2) Plate localization.

(3) Character segmentation.

(4) Centroid method.

3.2. Converting the Image into Gray Scale. All frames that are passed to the image recognition function are converted into gray scale using standard functions available in OpenCV. Gray scale image, by definition, includes all the relevant information for license plate detection and removes any information that may be irrelevant, including color and other characteristics, while retaining the brightness level of the image. Image processing method does not require the additional characteristics of the image. Moreover, the gray scale image is 8-bit monochrome, which equates to less power utilization during detection phases.

3.3. Plate Localization. To locate the plate, combinations of standard image derivative methods described above (Hough transformation, pattern matching, etc.) are used. In addition, to achieve the desired efficiency and quality of the detection, the following filters that are the standard OpenCV functions are applied [14, 15]:

(i) *Median blur*: Since the overall quality of the images produced by the Glass camera is significantly low, this is a preprocessing step done to improve the quality of the detection. A median filter works by replacing each pixel in the image with the median value of the neighboring pixels in all dimensions.

(ii) *Adaptive thresholding*: Adaptive thresholding usually takes a gray scale or color image as an input in the simplest implementation and outputs a binary image representing the segmentation. For each pixel in the image, a threshold has to be calculated. If the pixel value is below the threshold it is set to the background value; otherwise, it assumes the foreground value.

(iii) *Finding contours*: Contouring provides a key benefit in highlighting the shape of an object. As the contour of a given pattern is obtained, its distinctive characteristics will be observed and used as features, which will then be used in pattern classification. Thus, accurate extraction of the contour will produce more accurate features that will increase the chances of correctly classifying a given pattern.

3.4. Character Segmentation. Character segmentation involves chopping the additional region around the area where the license plate is detected. This, however, is not a simple mathematical operation; rather an array of regions where the characters are most likely to be found are collected. We used the Kohonen network pattern matching to achieve better accuracy. To ensure that the characters that are to be recognized are in correct order (e.g., for a license plate containing number 7193, order 7-1-9-3 is ensured using the centroid technique; it ensures that 7193 appears as 7-1-9-3 and not 9-3-1-7 or any incorrect order).

3.5. Centroid Method. The process of character segmentation is not definitive and the results that are often returned from this function are not in the correct order of the characters in the plate. The centroid technique designed for gPark ensures that the characters are in order, by calculating a point at the center of the plate and the distance from that particular point to the rest of the regions of the plate. The regions to the far left correspond to the first character and those to the far right correspond to the last character, in order.

We used a pretrained Kohonen encoded file containing the pattern of the characters released on open source domain. The application simply matches the results of the character segmentation results above with the contents of the data in the Kohonen file. This helps us create multiple customized training sets for different countries, and ensure that different types of plates can be detected.

During the initial testing of the application, it was found that the recognition of the characters was not as accurate when it comes to similar characters. For example, 8 could be mistaken for B or 5 could be mistaken for S, 0 could be mistaken for for O, and so forth. In order to solve this problem, we implemented few custom localization methods to avoid similar issues. This is applied as a calibration function in the application. An example of the set of customization rules is the following:

(i) If an alphabet appears after a numerical value, it is always a numerical value and is mapped to the corresponding numerical value.

(ii) The first character must always be an alphabet.

(iii) An alphabet can never appear in between two numeric values.

4. Testing Results and Discussions

The functionality of the system was tested with respect to the detection rate and accuracy of the ANPR algorithm. We carried out the testing of our application on real-life actual plates under two different outdoor conditions: natural lighting and low lighting (backlit).

When the working of the algorithm on Glass was tested in natural bright daylight condition, it was found that, due to the high contrast between the license plate and the surrounding, the ANPR algorithm was not able to detect the edges of the plate accurately in some cases. Out of 40 tests, only 22 were successful, giving us the detection accuracy of only 55%.

The testing procedure was repeated in low-light conditions. It was seen that the accuracy was about 75%—much better when there is backlit lighting. The main reason is that there is a balanced contrast between the vehicle plate and the surroundings, leading to clearly defined license plate edges and hence higher detection accuracy.

One possible explanation of low detection accuracy in sunlight is the poor Google Glass ISO performance. ISO performance is a common term used in photography, which defines the level of sensitivity of a camera lens. In almost every modern-day camera, users are allowed to control the ISO settings of the camera. The ISO setting of Glass ranges from 60 to 900 and it is automatically controlled by the operating system running on the Google Glass. This, however, at this point, is not controllable manually, since it is still in its prototype stages. Due to this factor, on bright days, the contrast of the images is too high and thus the Glass camera produces low quality image that does not contain enough information to distinguish the plate from the surrounding regions—giving low detection accuracy results.

5. Conclusions

Wearable technologies can play an important role in improvements in present-day vehicle parking management systems. We have described the design of a parking management system that uses Google Glass and smartphone along with the remote databases servers. We have designed and implemented the application on the Glass and tested it in real-life parking situations. The number plate detection accuracy of the application can reach up to about 75%. We believe that the accuracy can still be enhanced with the fine tuning of OCR algorithms used and better hardware in the future versions of the Google Glass.

Competing Interests

The author declares that they have no competing interests.

Acknowledgments

The author would like to thank the following students who participated actively in the design and implementation of the prototype of this project: Benna Iqbal, Maryam Hassan, and Samina Abdul Rahman.

References

[1] J. Tang, *Beginning Google Glass Development*, Apress, New York, NY, USA, 2014.

[2] The Mirror API, https://developers.google.com/glass/develop/mirror/index.

[3] S. Mann, "Through the glass, lightly," *IEEE Technology and Society Magazine*, vol. 31, no. 3, pp. 10–14, 2012.

[4] S. Draghici, "A neural network based artificial vision system for licence plate recognition," *International Journal of Neural Systems*, vol. 8, no. 1, pp. 113–126, 1997.

[5] C. Nikolaos, "License plate recognition from still images and video sequences—a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 377–391, 2008.

[6] B. Chen and H. H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 485–497, 2010.

[7] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic License Plate Recognition (ALPR): a state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311–325, 2013.

[8] J. Tian, R. Wang, G. Wang, J. Liu, and Y. Xia, "A two-stage character segmentation method for Chinese license plate," *Computers & Electrical Engineering*, vol. 46, pp. 539–553, 2014.

[9] M. Chalamish, D. Sarne, and R. Lin, "Enhancing parking simulations using peer-designed agents," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 492–498, 2013.

[10] OpenCV, http://opencv.org/about.html.

[11] J. Desautels, *Google Glass Forensics*, 2014, http://smarter-forensics.com/wp-content/uploads/2014/06/Google-Glass-Forensics.pdf.

[12] Glass Development Kit, https://developers.google.com/glass/develop/gdk/.

[13] T. Nguyen, D. Nguyen, and P. Nguyen, "UIT-ANPR: toward an open framework for automatic number plate recognition on smartphones," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (ICUIMC '14)*, ACM, Siem Reap, Cambodia, January 2014.

[14] J. Kim and H. Jun, "Implementation of image processing and augmented reality programs for smart mobile device," in *Proceedings of the 6th International Forum on Strategic Technology (IFOST '11)*, pp. 1070–1073, Harbin, China, August 2011.

[15] J. Kim, Y. Han, and H. Hahn, "Character segmentation method for a License plate with topological transform," *Transactions of World Academy of Science, Engineering and Technology*, vol. 56, pp. 39–42, 2009.