Crowdsourcing Automobile Parking Availability Sensing Using Mobile Phones

Jesus Villalobos, Bereket Kifle, Derek Riley and Jesús Ubaldo Quevedo-Torrero
University of Wisconsin Parkside
Kenosha, WI 53141
kifle001@rangers.uwp.edu
villa054@rangers.uwp.edu
rileyd@uwp.edu
quevedo@uwp.edu

Abstract

A lack of reliable knowledge about automobile parking availability in areas such as schools, work, or major cities wastes time, energy, and fuel as people try to find available parking spaces. Real-time parking monitoring phone applications exist, but keeping accurate, reliable parking availability information proves to be a difficult task due to the unreliability of real time information, especially in less densely populated areas. In this paper, we present a parking monitoring system that uses crowdsourcing in combination with mobile phone sensors to provide accurate, reliable real-time parking availability information. We present a study of the use of the application on a university campus to demonstrate its effectiveness.

1 Introduction

Throughout the world, automobile parking space is a resource that is often scarce due to the popularity of automobiles in urban areas. Often, drivers have to spend time in addition to fuel looking for a spot, which could lead to lateness as well as additional air pollution [11]. Improving access to real-time parking availability information has the potential to improve lot utility, energy efficiency, and time savings. Real-time parking information is a valuable commodity in situations where alternative parking locations are available. A survey [12] showed that out of over 480 drivers, 30% changed their intended parking destination after seeing road signs indicating open parking elsewhere. The ability to access information about the availability of spaces in parking lots can change rapidly, so it is ideal to provide this information on a mobile phone application.

Monitoring and logging parking space availability is a difficult task to perform for several reasons. Systems exist that use wireless sensors to monitor individual spots, but these systems can be costly to maintain and return erroneous results if they aren’t calibrated regularly [3]. Mobile applications using QR codes at every parking space are a cost efficient alternative, but these require universal participation of users, which can lead to unreliable results, and is impractical for large parking lots [2]. Using crowdsourcing to determine parking availability is a promising solution to collecting parking availability data because of its potential to be deployed on any size scale and at a low cost. Reliability of information from a tracking system is critical because if users don’t trust the reliability of the information, it will undermine the participation in the application. This is especially important when the application relies on crowdsourced data because accurate information can easily be outweighed by malicious or erroneous inputs and can make the application unreliable [6].

In this paper, we present a prototype parking monitoring system that uses crowdsourcing and mobile phone sensors to provide more reliable parking availability information for users to find parking in a faster, more efficient manner. A mobile phone app uses GPS to determine the users’ location and prompts them to vote to help determine the congestion of the lot. If the user chooses not to vote, the app will continue to track their GPS location to determine whether they are leaving or entering the lot and what specific zone of the lot they are in using vector mapping. All crowdsourced information is combined in an algorithm to provide a prediction of the fullness of each zone of the parking lot. To reduce the influence of crowdsourcing data manipulation, expert data is also collected by an authority figure such as parking officials or the police department. These authority figures can use a hidden feature on our app to directly insert reliable data to improve the overall quality of data gathered.

2 Background

In recent years, crowdsourcing has been shown to be a powerful tool that can be used to solve complex problems. It is different than traditional computational methods because it integrates aspects of `human computation’ into computer algorithms to create new approaches to handle problem solving. Google’s CAPTCHA service leverages this fact to use people to recognize a warped text line from scanned documents [8]. It is difficult for a computer to recognize warped letters while humans typically identify the underlying characters easily. Despite being a relatively new concept, crowdsourcing has spread with the ever-growing availability of mobile computers.

Transportation-based services are a frequent topic in crowdsourcing applications for mobile devices. In most of these approaches, several algorithms have been designed to determine parking lot congestion [2][3][4][5][6][7]. There have been approaches using QR-Codes for users to log incoming and outgoing parking as described in [2], wireless sensor networks as listed in [3], and GPS in [2][3][6][7]. These approaches are all error prone because they rely on a single source of data and this can lead to inconsistencies such as GPS inaccuracies, network failures, device failures, and malicious manipulation of crowdsourced data. Other solutions to the parking lot congestion problem have relied mainly on crowdsourcing. For example, in TruCentive, described in [7], the parking problem is presented as a game, and in their approach a reward system is used to motivate people to input data on parking coupled with the ability to reserve free parking spots. However, applications that rely heavily on input from users collapse if not enough users participate [6].

Establishing a strong motivation for using an application is a key component to ensure optimum crowd participation and high quality of crowdsourced data. Three ways to motivate a crowd are entertainment, rewards, and benefits. An experiment was conducted in an online labor market that measured the effectiveness of a collection of social and financial schemes for motivating workers to conduct a qualitative, content analysis task. The experiment showed that financial incentives combined with peer reviews produced more accurate participation from workers [9].

3 UW-ParkAssist

The UW-ParkAssist application is designed to provide users with real-time information about parking availability in UW-Parkside's campus lots. Users are motivated to use the application because the information the app provides will save the users' time as they look for a parking space, as long the data is accurate. A large portion of the students and staff at UW-Parkside are commuters, so parking space availability is a high priority for a large part of the community. To reinforce the reliability of the collected information we have included the ability for authorized authorities such as the police department to insert expert data to override incorrect data.

UW-ParkAssist combines an active and passive solution of obtaining data from the user to reduce the reliance of the application on exclusively crowdsourced data. UW-ParkAssist combines data collected passively using vector-based location mapping on users' phones with crowdsourced information actively provided by users related to their observed density of parking. We create a grid using vector based mapping, with boundaries in certain areas of the parking lot to record coarse user movement information, simplifying the parking lot zones.

The passive portion of our application uses vector-mapping to improve user participation rates. A vector map is a map that uses vector data composed of points and lines that are geometrically related. Vector data is valuable in mapping because it minimizes the need to track exact GPS coordinates and headings [1]. In UW-ParkAssist, we used nodes in a similar manner to the walking application [1], but instead we created zones using various nodes to split the lot into sections, as shown in Figure 1. The red, smaller boxes in the figure represent vector zones that are hidden from the user. These vector zones determine whether a user is entering or leaving the parking lot. The black lines indicate how the parking spaces within the main lot are divided into six zones.

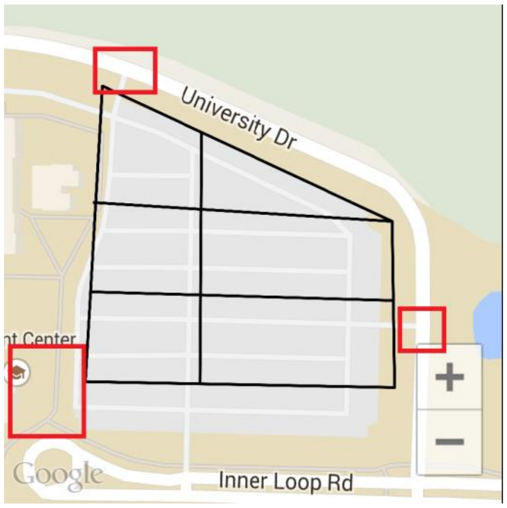
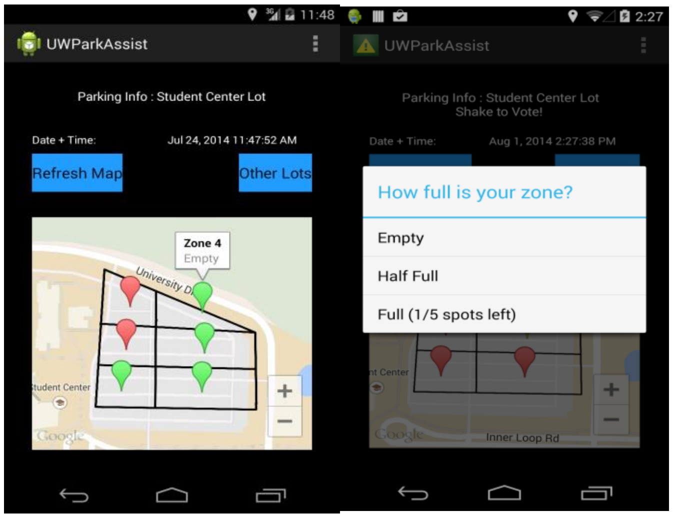


Figure 1: Example parking lot in the application. Red boxes indicate hidden vector zones used to determine arrivals and departures. Black lines indicate zones for lot sections.

The user interface implemented in the app consists of a map fragment that displays an overhead map of the main, as well as a label asking the user to shake the device in order to vote. The map contains markers that divide the parking lot into zones to show which areas of the lots are full and which ones are empty. The lot is separated into sections defined as zones shown in Figures 1 and 2. We divided the lot into zones to aggregate the parking information instead of overwhelming a user with detailed space-by-space information. These zones are used to help track when the user enters and leaves the parking lot.



Parking Info : Student Center Lot

Date + Time: Jul 24, 2014 11:47:52 AM

Parking Info : Student Center Lot
Shake to Vote!

Date + Time: Aug 1, 2014 2:27:38 PM

How full is your zone?

Empty

Half Full

Full (1/5 spots left)

Figure 2: Left: Main screen for UW-ParkAssist showing the six parking zones. Right: Voting popup after an accelerometer detects movement.

To collect active user location data, the application senses when a user is leaving their car using the accelerometer. When the phone is shaken, whether it's from unplugging the mobile device from a charger or gathering up items needed for class, the accelerometer registers the motion so that UW-ParkAssist can determine in which zone the user is located using the GPS sensor. After the zone is determined, a pop-up window appears, asking the user to vote on how full their zone is to encourage them to contribute to the crowdsourcing portion of the application, as shown in Figure 2. If the accelerometer does not detect the shaking when leaving a car, the GPS sensor can be used with a timer to track motion and figure out where the user parked.

UW-ParkAssist also uses expert data from the UW-Police to reinforce the status of the parking lot and to improve data quality. The application has a hidden button leading to a screen where parking officials or the police can manually adjust the information in the database. A vote from one of these authority figures overrides the current status of a zone, and authorities also have the ability to mark all zones as being full or empty. This feature allows authorities to use the app when they are in the lots on patrol routes or when they view the lots from security cameras. These features are important to obtain the most reliable data possible and avoid user error, inconsistent data, or manipulation of the application.

All of the information on each parking lot zone is stored inside a MySQL database. The database interfaces with a RESTful API, reachable by any user with an Android device, access to the internet, and the application. The application uses a REST client which handles the create, read, update, and delete operations performed on the database. We used a RESTful web service due to its ease of use for development and implementation[13]. When the mobile application starts up, zone data is retrieved from the database and parsed into arrays. These arrays are then used to populate the map fragment displayed on the main screen.

If a vote is registered or one of the vector zones are entered, the user’s zone location and vote are sent back to the database via the REST client. The vote is validated using GPS, and votes only reach the database if the user is actually in the parking lot when the accelerometer is triggered. In both the inactive, passive scenario and the active, voting scenario (where the user shakes the device in order to vote on parking lot density), the device also stores the last known location using GPS, along with which marker the phone was closest to last. Since a large amount of user participation is not guaranteed, the vector mapping, passive portion of the application will normally be less accurate than the active portion. Because of this, we rely on voting along with expert data to override the passive aspect of the application.

4 Crowdsourcing

The crowdsourcing component of UW-ParkAssist is designed to be straightforward to allow users to understand how their information is used and where the data comes from, which is intended to help increase user participation. When a user votes on the density of a zone, the application logs the vote in a table within the database. The database is analyzed by a voting algorithm that assesses the votes and determines the result that the application should display for a given zone (empty, half full, full).

Our voting algorithm is divided into 4 main components: input data, input voting, output data, and output voting as shown in Figure 3. The data portion of the algorithm refers to the data objects sent back and forth between the device and the database. The voting portion refers to the actions taken based on the data objects. For our application, the input data is exact, meaning that all votes received have inflexible values (0 for empty, 1 for half full, and 2 for full) instead of flexible "neighborhoods" [10]. The input voting is also pre-set, which means that votes cannot be adjusted or turned into variables afterwards. The output data displays a consensus of the crowdsourced votes.

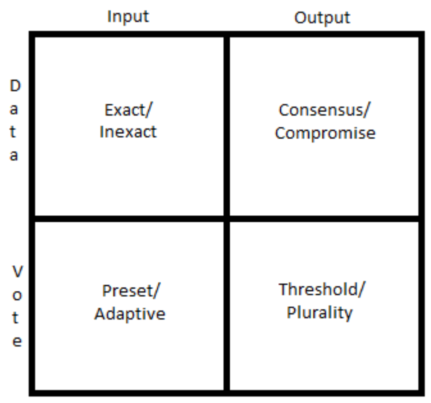


Figure 3: Classification for Voting Algorithms [10].

In our initial implementation, we set *w* and *T* to 3 for our experiments. When 3 votes for the same density are recorded, the color of the marker is changed and the votes, along with the number of spots available, are reset in the database to preset values. Votes made by an authority figure have the same weight as 3 common user votes, meaning that a zone can be reset with a single vote from an authority figure.

5 Experimental Design

First, we wanted to evaluate the crowdsourcing voting aspect of our application, so we designed an experiment to determine how people would vote on parking lot density they perceived. The participants were shown six pictures of different views of zones in the parking lot. We found that empty and full zone pictures garnered the most consistent votes. Participants voted appropriately 100% of the time that the zones were either empty or full, respectively. When shown pictures of lots that were not empty or full, participants voted that a zone was half full 85% of the time, while 15% voted that the zone was empty. These discrepancies are likely due to overall perceptions of the users, but would not adversely affect the usefulness of the crowdsourced information.

Our next experiment was constructed to measure the accuracy of parking density information generated by UW-ParkAssist. We simulated a user by going out into the parking lot and the randomly generated densities to simulate how a driver would interact with the application. The densities were determined as empty, half full, or full with a random number generator. We then focused on users entering and leaving one zone that held 52 spots, to judge how the vector-mapping aspect interacted with the active, voting aspect. The zone started off as half full, and we simulated 15 drivers parking in that zone with another 15 drivers leaving.

The drivers were simulated individually to see how each one would vote based on the latest information available. The fifteen drivers arriving to the parking lot were separated from the fifteen leaving, to simulate the rush of people coming for and leaving from class. We separated the first fifteen drivers from the second by only simulating drivers entering the parking lot and parking in that zone initially. Afterwards, we simulated fifteen drivers leaving that zone of the parking lot. For both groups of drivers, we had 6 of the users actively voting within the zone, while the other 9 users simply walked either to or from the zone. We simulated 6 users who actively voted in order to meet the requirements for the voting algorithm.

The combination of voting and passive vector-mapping resulted in a correct change from half full to full after the first fifteen users entered the zone. Unfortunately, the fact that the half full vote reset the spots left in the database meant that in order for the zone to be recognized as full, more active participation was needed. The zone only changed to full after the last of the 15 simulated drivers entered and parked in the zone, which possibly could have taken too long for a user who is not constantly refreshing the map. In the case where users were leaving school, the application performed properly by changing the zone back to yellow.

For our final experiment, we collected data on how long in took find parking during the peak hours of the school day. On average it took 4 minutes and 35 seconds to find parking at peak times. We then simulated how long it would take to find a parking spot using the application. On average, it took less than 45 seconds from entering the lot to parking in a spot, regardless of which entrance was used and which zone was open.

6 Conclusions and Future Work

Crowdsourcing is a promising approach for determining parking availability, although complications accompany this solution such as user participation, legitimate data from users and user error while using the application. This paper demonstrates a working mobile application in where we do not only rely on crowdsourcing but on expert data to improve the reliability of the information. We created a way to facilitate finding a parking spot by designing a parking monitoring system that uses crowdsourcing in combination GPS data, vector-based mapping, mobile sensors, and the expert data from a reliable source. In our case the UW-Police is the source for the expert data. For future work, one of the possibilities is collecting data from the parking lot in a longer time span and including historical data. The app then could then inform users about certain times of day where the parking lot is most often full, as well as compare the historical data with the most current data and produce density probabilities of the lot.

References

[1] Ookura, Hikaru, Hiroshi Yamamoto, and Katsuyuki Yamazaki. "Development and evaluation of walking path estimation system using sensors of Android device and vector map matching." *Information Networking (ICOIN), 2012 International Conference on*. IEEE, 2012

[2] Bechini, Alessio, Francesco Marcelloni, and Armando Segatori. "A mobile application leveraging QR-codes to support efficient urban parking." *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*. IEEE, 2013.

[3] Yang, Jihoon, Jorge Portilla, and Teresa Riesgo. "Smart parking service based on wireless sensor networks." *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012.

[4] Kun-Chan Lan & Wen-Yuah Shih (2014) “An indoor location tracking system for smart parking”, International Journal of Parallel, Emergent and Distributed Systems, 29:3, 215-238,

[5] Chen, Xiao, Elizeu Santos-Neto, and Matei Ripeanu. "Crowdsourcing for on-street smart parking." *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*. ACM, 2012.

[6] Hoh, Baik, et al. "TruCentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services." *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012.

[7] Davami, Erfan, and Gita Sukthankar. "Evaluating trust-based fusion models for participatory sensing applications." *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[8] Yuen, Man-Ching, Ling-Jyh Chen, and Irwin King. "A survey of human computation systems." *Computational Science and Engineering, 2009. CSE'09. International Conference on*. Vol. 4. IEEE, 2009.

[9] Shaw, Aaron D., John J. Horton, and Daniel L. Chen. "Designing incentives for inexpert human raters." *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. ACM, 2011.

[10] Parhami, Behrooz. "Voting algorithms." *Reliability, IEEE Transactions on* 43.4 (1994): 617-629.

[11] Arnott, Richard, and John Rowse. "Downtown parking in auto city." *Regional Science and Urban Economics* 39.1 (2009): 1-14.

[12]Mike McDonald and Kiron Chatterjee. “VMS in urban areas - Results of cross-project collaborative study.” Technical Report TR1101 D3.3.1-a, Transport Sector of the Telematics Applications Programme (T-TAP), March 2000.

[13]What Are RESTful Web Services? - The Java EE 6 Tutorial. (n.d.). Retrieved August 1, 2014, from https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html