논문 20-45-07-20 The Journal of Korean Institute of Communications and Information Sciences '20-07 Vol.45 No.07
https://doi.org/10.7840/kics.2020.45.7.1277

A Novel Connected Vehicle-Based Parking Space Guidance System

Yugesh KC◆, Chang Soon Kang*

ABSTRACT

This paper proposes a novel connected vehicle-based parking space guidance system, which improves the effectiveness of on-street (roadside) parking by detecting, updating, and processing available parking spaces in real time. The proposed system consists of a detecting device installed in the car, a cloud server, and a mobile application connected to LTE cellular networks, in which the server examines the correctness of available parking spaces found by the detection device, selects feasible parking locations (FPLs) based on the user's driving and walking distances, and provides the FPLs to the users requesting parking locations. Furthermore, with the mobile application, users can select a feasible parking location from the FPLs and navigate to the desired parking location. In particular, the proposed system resolves unnecessary navigations to the parking spaces intercepted earlier by other drivers and also provides parking facilities when FPLs are unavailable. We developed the system as a prototype and evaluated its performance through field and laboratory trials. The evaluation results show that the system effectively provides FPLs to users looking for parking spaces. With the application of the system to a smart city, it is expected to significantly alleviate the problems caused by cars roaming in search of parking lots.

Key Words : Connected vehicle, Internet-of-Things, Intelligent transport system, On-street parking, Smart city

Ⅰ. Introduction

Most objects around us will soon be connected with communication networks in the Internet of Things (IoT) paradigm, a term that was initially introduced in 1999 by Kevin Ashton[1]. The IoT consists of a device domain for measuring information using actuators, sensors, or embedded systems, a network domain for transmitting the measured information to a server, and an application domain for analyzing and storing the information on the server[2]. Recently, there has been a substantial increase in connected objects in the networks to create a smart environment. "IHS Technology" has forecasted that the market of the IoT may reach up to 75.4 billion devices by the end of 2025[3]. Many applications will be impacted by the emergence of the IoT, such as healthcare, emergency services, defense services, smart agriculture, smart transportation, connected vehicles, etc.[4].

Connected vehicles are wireless connectivity enabled vehicles that are capable of communicating with their external and internal environments. Connected vehicles support such interaction as

※ A preliminary version of this paper appeared in IoTAIS 2019, November 5-7, Bali, Indonesia [23]. This version includes a new idea with the extension of the study in the conference paper. This research was financially supported by Changwon National University in 2020.

◆ First Author : Department of Eco-Friendly Offshore Plant FEED Engineering, Changwon National University, Republic of Korea, uges.kk@gmail.com, Student member

* Department of Information and Communication Engineering, Changwon National University, Republic of Korea, cskang@changwon.ac.kr, Life member

논문번호 : 202004-084-D-RE, Received April 8, 2020; Revised May 1, 2020; Accepted May 1, 2020

vehicle to vehicle, vehicle to on-board sensors, vehicle to road infrastructure, and vehicle to the internet[5]. An industry report of Telefonica has predicted that the percentage of internet-integrated vehicles will jump to 90% by 2020[6]. Bringing wireless connectivity to vehicles enables several promising solutions for accident prevention, self-diagnosis, alleviation of traffic congestion, smart parking, and so on [7].

The increase in vehicles in urban areas has created a shortage of parking spaces, which has become a common problem. This problem causes undesirable issues such as waste of time, air pollution, energy consumption, and even a significant cause of traffic congestion in commercial and residential areas[8]. According to recent research[9], 30% of cars wander around looking for parking places, which takes about eight minutes. Moreover, another study[10] demonstrated that traffic congestion not only causes more travel time and energy consumption, but it also increases the emission of pollutants like carbon monoxide (CO), carbon dioxide (CO_2), nitrogen oxide (NOx), and other pollutants associated with fine dust. The construction of new parking facilities to mitigate parking-related problems is an unsuitable measure because it requires a significant expenditure, and the spatial resources are limited in cities, and thus most cities consider the on-street parking spaces as a prime choice[11].

1.1 Related Works

Several research studies have introduced solutions to overcome the problem caused by the shortage of parking spaces; most of the studies have focused on off-street parking. Wang and He[12] proposed a smart parking system that uses vibration sensors and light sensors to detect occupancy of parking spots, in which the sensors transmit the sensed data to a server using the Zigbee protocol (IEEE 802.15.4), and a user can make a reservation via mobile application. The proposed system proved its usefulness using a simulation and showed that the proposed parking system had the potential to alleviate traffic congestion by reducing the number of vehicles cruising for parking space. In addition, other studies on parking systems focused on the enhancement of off-street parking by implementing different technologies: the authors in [13] proposed a parking system with ultrasonic sensors and a cloud-based IBM Message Queuing Telemetry Transport (MQTT) server, and the authors in [14] used an ultrasonic sensor for the parking system in combination with Bluetooth communication to check occupancy and validation of users. Furthermore, the authors[15] used an image processing scheme to detect the occupancy of parking spaces; similarly, other smart parking research has been done using different sensing techniques; smart cameras were used for the parking system in [16], whereas ferromagnetic sensors were used in [17].

Furthermore, there has been growth in the number of on-street parking systems due to the limited spatial resources of the city. One of the most well-known systems is the SFpark[18], in which 11700 magnetometer sensors and 300 pole-mounted mesh nodes were installed to build 8000 parking spaces with a total budget of 46.2 million dollars from 2009 to 2014. The FastTrack[19] and GEOmii [20] are similar projects with different sensing techniques, namely, magnetic and magnetometer sensors, respectively. These systems provide such benefits as easily find parking spaces, reduced congestion, lower parking rates, and the issuance of fewer parking tickets; however, they require higher expenses for deployment and maintenance.

All of the previous studies[12-20] applied a field sensor detecting method to smart parking systems, in which sensors were pre-deployed at parking locations to detect the occupancy of parking spaces. Although this method provided higher accuracy for commercial services, applying it to roadside parking would be inefficient in terms of cost and maintenance. On the other hand, a car sensor detecting method was proposed to overcome the disadvantages of field sensor detecting schemes, in which sensors installed in a car collect parking occupancy information. Mathur *et al.*[21] published a roadside parking system, the ParkNet, using a car sensor detecting method, in which the system used

ultrasonic rangefinders designated to collect the parking occupancy information when a car passed by the roadside. The authors constructed a parking map for the ParkNet based on the collected data from the field trials that lasted over 30 days and implemented a fingerprinting approach to improve the accuracy of the parking location. In addition, the authors in the study[11] proposed a smart parking system that used an approach similar to the ParkNet to collect parking occupancy information; the only difference between the two studies was the technology that was used, in which the proposed system consisted of the Raspberry Pi, a cellular modem, a GPS receiver, and an ultrasonic rangefinder that uses a supervised learning-based algorithm for detecting occupancy and a map matching technique to correct the location errors. Another study[22] used a computer vision technology with a machine-learning approach to detect on-street parking.

However, these systems in [11], [21], and [22] have disadvantages such that the systems are used mainly for on-street parking; the vehicles may redundantly upload parking space information when multiple monitoring vehicles pass by the same parking space available. In addition, the systems could not update the parking occupancy information in real time, consequently, thereby causing unnecessary navigations to the parking spaces unaware of any interceptions created by other drivers. To resolve the above drawbacks of the parking systems, another study[23] presented a connected car-based parking location service system suitable for both on-street parking and off-street parking, in which a detecting device installed in each car uploads the information it finds, such as a parking location, car id, and free space width to a server. The system provides parking location information near to the driver's intended destination via a mobile application.

1.2 Contributions

In this paper, we propose a car sensor detecting method-based on-street parking space guidance system that includes a new idea with extension and implementation of the study[23], in which the system consists of a detecting device for finding available parking spaces, a cloud-based server, and a mobile application. Our proposed system detects available parking space information (APSI) and examines the correctness of the information; it also adopts an application programming interface (API) that processes the APSI to select feasible parking locations (FPLs) based on the user's driving and walking distances. With the mobile application, users can select an FPL and navigate to their desired parking location.

Furthermore, the proposed system can provide parking facilities to users in the absence of available on-street parking spaces and handles the interceptions of parking spaces from other users by detecting and updating a specific available parking space in real time. In particular, the proposed system increases the possibility of finding available parking spaces by improving the effectiveness of on-street parking. We developed the system as a prototype and evaluated its performance through field and laboratory trials, in terms of four performance measures: the accuracy of parking locations, the average time for collecting APSI, the average time for providing FPLs, and the average time for selecting optional FPLs. Note that in this paper, the on-street parking represents roadside parking, and the off-street parking indicates parking facilities, including private parking and public parking.

1.3 Organization

This paper is organized as follows: Section II describes the design of the proposed system, including the requirements, overview, and architecture of the system. Section III discusses the tasks of API for parking locations, and Section IV presents the performance evaluation of the system through field and laboratory trials. Finally, Section V concludes this paper and suggests further research.

Ⅱ. Design of the Proposed System

2.1 System Overview

The proposed on-street parking space guidance system considers the requirements described in Table 1.

The proposed system consists of a detecting device (DD) installed in the car, a cloud-based server, and a mobile application for users. The DD consists of ultrasonic sensors that detect available parking spaces, which is uploaded later to the server along with additional information collectively termed as APSI. The APSI includes available parking spaces present in the front and back sides of a vehicle, the unique identification (ID) of the DD, and GPS coordinates of the available parking spaces. When the server receives the APSI, it inspects the correctness of the information before storing it, and in addition, the server processes the stored APSI and responds to the user's request for feasible parking locations (FPLs). In addition, the server suggests parking facilities near the user's destination only when the FPLs are not available. Finally, the user can navigate to the desired parking location via a mobile application. Here, the FPLs indicate on-street parking locations *available near the user's destination*, whereas the *available parking locations* are on-street parking locations where cars can be parked, regardless of the user's destination.

Table 1. System requirements.

System requirements	Details
Detecting available parking spaces.	The available parking spaces can be detected with ultrasonic sensors in the car.
Filtering parking-related information.	A server checks the correctness of the parking-related information collected from a detecting device in the car before inclusion into the database.
Providing feasible parking locations (FPLs).	The server processes the stored information to provide the FPLs on the user request.
Navigation service.	With the help of a mobile application, the user can navigate to the desired FPL.

2.2 System Architecture

We designed the proposed system considering the system requirements, in which the system consists of a DD in a car, a cloud-based server, and a mobile application, as depicted in Figure 1. Initially, the DD measures available parking spaces and uploads the information (i.e., the APSI) to the server; the information is processed and stored in the server. After that, a user can ask the server for FPLs and navigate to the desired FPL using the mobile application. The sequence flow of the system is shown in Figure 2, and the details of the flow are described below.

The DD includes the available parking space information inside the Transmission Control Protocol (TCP) packet, as shown in Figure 3, and transmits

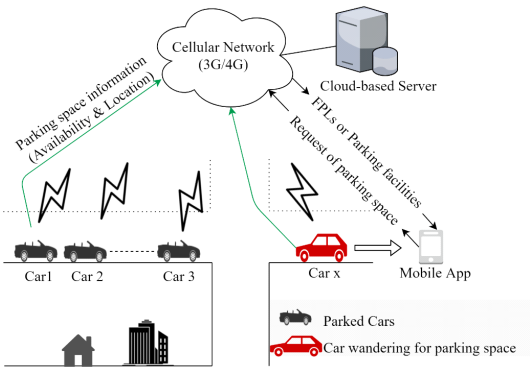


Fig. 1. Architecture of the proposed parking space guidance system.

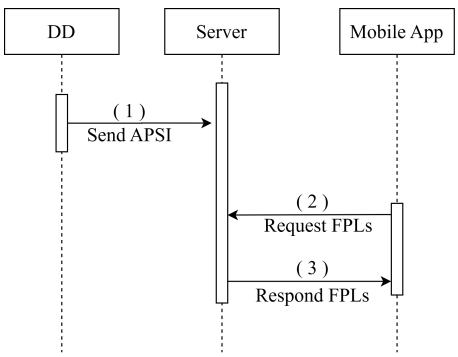


Fig. 2. Overall sequence flow of the proposed system.

1. DD to Server

HEAD	TCP Data (Available parking space information: APSI)						
	...	Car ID	GPS coordinate	Available front side width	Available backside width	…	END
@	…	[value]	[value]	[value]	[value]	…	#

2. Mobile Application to Server

HEAD	TCP Data (User's request)						
	….	User ID	User's current location	User's destination location	Length of user's car(m)	…	END
@	….	[value]	[value]	[value]	[value]	…	#

3. Server to Mobile Application

HEAD	TCP Data (Response from the server)					
	…..	Coordinates of parking location (Option 1)	Coordinates of parking location (Option 2)	Coordinates of parking location (Option 3)	…	END
@	…..	[value]	[value]	[value]	…	#

Fig. 3. Message formats of the request and response between the DD, server and mobile application.

it to the server through a Long-Term Evolution (LTE) network. Similarly, the mobile application includes the request parameters, such as the user's ID, the car's length, and the user's current and destination locations in a TCP packet and sends a request message to the server for feasible parking locations. Finally, the server responds with the FPLs. The message formats of the request and response between the DD, server, and mobile app are shown in Figure 3.

2.2.1 Detection Device (DD) for Available Parking Spaces

The DD is designed with a controller (e.g., Raspberry Pi), a GPS receiver for the parking location, two ultrasonic sensors for measuring the distance between cars, and a cellular (e.g., Long-Term Evolution: LTE) modem as shown in Figure 4. Each DD is installed in a car in which all of the requisite hardware is inside except the sensors, which are installed at the front and back of the car because cars are usually parked in a linear order in on-street parking, as shown in Figure 5. The LTE modem is a prime choice as it provides ubiquitous communication, which is essential for the proposed system.

The DD monitors the states of a car, such as moves or stops, and identifies the state as 'parked' if the car stops for a predefined threshold time; otherwise, it recognizes the state as a move. The GPS coordinates are required to ensure that the vehicle has stopped because the coordinates could be different even if the vehicle is stopped. Therefore, the controller (Raspberry pi) retrieves new GPS coordinates from the GPS receiver every 30 seconds and computes the distance between the new and previously retrieved GPS coordinates using the haversine formula[24].

If the computed distance is smaller than a threshold distance for determining the movement (D_{mth}), the vehicle is considered as being stopped,

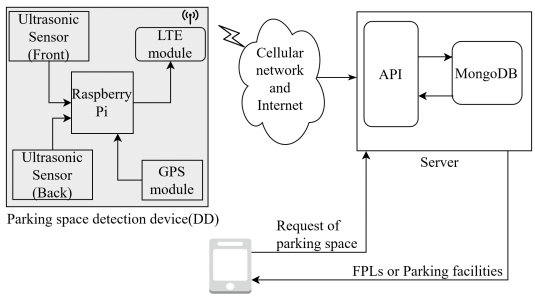


Fig. 4. Hardware configuration of DD in the proposed system.

(a) (b)

Fig. 5. Typical on-street parking: (a) commercial area (b) residential area.

and thus, the stop time is incremented by 0.5 min; otherwise, the car is considered to be moving, and the value of the stop time is reset to 0. Whenever the stop time exceeds the threshold time, the vehicle is identified as parked. Figure 6 shows the algorithm to identify the movement states of vehicles, that is, moving or parked.

Right after the movement state of the vehicle becomes parked, ultrasonic sensors of the DD periodically measure the distance available from the

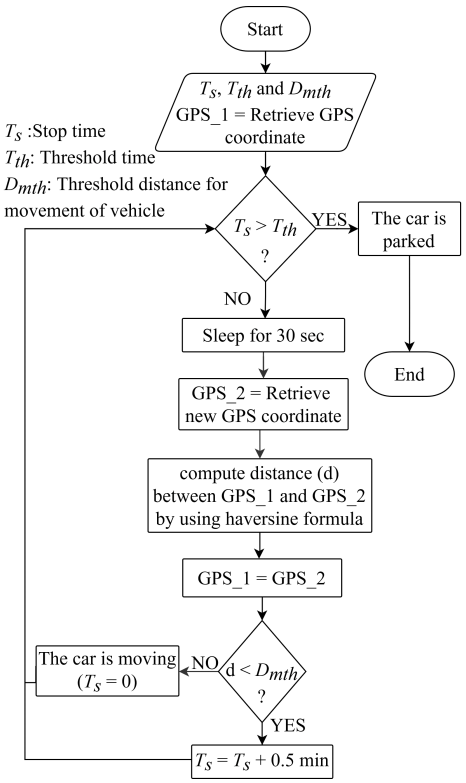


Fig. 6. Algorithm to identify the movement states of a vehicle.

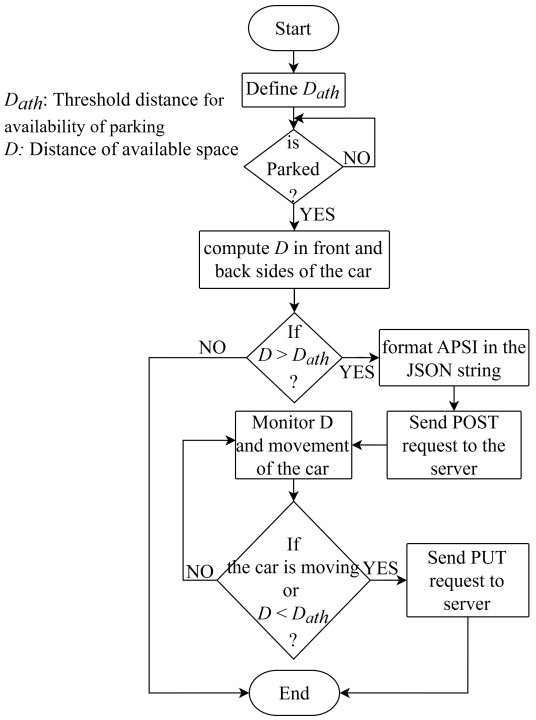


Fig. 7. Operation of the DD in the proposed system.

rear of the vehicle. Whenever the measured distance exceeds the threshold distance for determining the availability (D_{ath}) of a parking space, the APSI is formatted into the JSON string[25] and sent to the server via an HTTP POST request[1)][26]. After uploading, the DD further monitors the movement state of the vehicle and the available distance to ensure the availability of a parking space. If the DD detects any movement in the vehicle or if the measured distance available becomes less than the threshold distance, the APSI stored in the server is updated via a PUT request[2)]. Figure 7 depicts the operation of the DD in the proposed system

2.2.2 Cloud-based Server

The cloud-based server is a crucial part of the proposed system; it consists of the NoSQL database that stores the findings of the DD and API, where all of the logics are defined. We used MongoDB for our NoSQL database because it provides a free

1) The HTTP POST request sends data to the server.
2) The HTTP PUT request creates a new resource or replaces a representation of the target resource.

cloud-based service in a shared cluster[27]. Similarly, we developed a RESTful API and deployed it in a cloud-based server named Heroku[28]. Hereinafter, we use API instead of the developed RESTful API[3].

As soon as the server receives the APSI, the API validates its correctness before it is stored in the database. The validation of APSI involves two steps: (a) validation of vehicle locations and (b) validation of movement states. The detailed process for validating the findings of the DD is described in Section III. Moreover, the server is responsible for handling user requests: It retrieves and processes APSI to select FPLs and also provides the FPLs on user request, as shown in Figure 8.

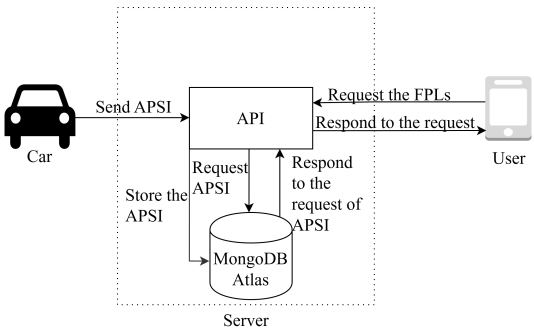


Fig. 8. Operation of the server in the proposed system.

2.2.3 Mobile Application

A mobile application is an interface through which a user can interact with the server in the proposed system. The mobile application performs tasks such as authentication, authorization, searching FPLs, selecting, and navigating to the user's desired FPL. Moreover, if any FPLs are not available, the mobile application can navigate the user to parking facilities near the user's destination.

Ⅲ. Tasks of the API

3.1 Validation of the Parking Location

When the API receives APSI from a car, it inspects the current location of the car and identifies

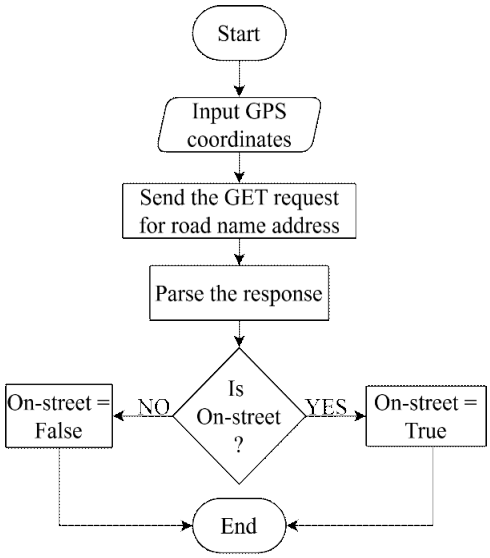


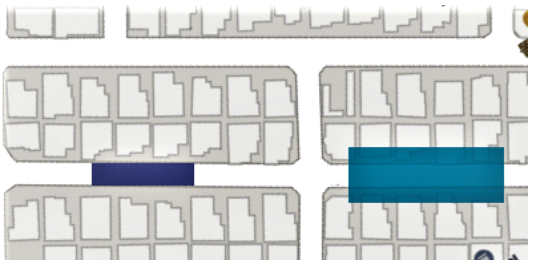
Fig. 9. Validation of a parking location in Scheme I.

whether the car is in an on-street or off-street location. To track geolocation, we used the reverse geocoding[4] of the *NAVER Maps API*, which requires an API key for authentication and mandatory query parameters such as a response format and GPS coordinates while performing a GET request. In addition, the reverse geocoding API returns information, including an administrative division (Dong) designated by the local Korean government and road name addresses for specific coordinates[29]. The primary version of the proposed system (denoted as Scheme I) performs a GET request for road name addresses, and the API parses the response to identify the parked location. Note that the reverse geocoding API returns the information based on an exact location of received GPS coordinates. Figure 9 depicts the validation process of a parking location used in Scheme I.

The validation algorithm for parking locations in Scheme I may be inaccurate if the API receives GPS coordinates that are located slightly outside of the streets because GPS coordinates are prone to have errors (Figure 10.a). The inaccurate validation may cause the system to suggest off-street parking facilities to users requesting for on-street parking

3) The RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

4) A reverse geocoding is the process that converts GPS coordinates into a readable address [29].



(a) (b)

Fig. 10. Parking locations: (a) on-street parking (b) specified on-street parking with wide boundaries.

spaces. Therefore, we improved the algorithm of Scheme I to minimize the impact of GPS errors (denoted as Scheme II). In particular, Scheme II specifies the on-street parking locations with wider boundaries, namely, an extra width on each side of the street (Figure 10.b) and stores it in the server with its 'Dong' address. (Note that the 'Dong' is an administrative unit of a sub municipal level in a city of South Korea.)

Later, the API requests the *reverse geocoding API* for an administrative division, which is parsed to identify the 'Dong' of a parked car. Afterward, the API retrieves specified parking locations that are located at the 'Dong' of the parked car and validates the parked location of a car as on-street only if the

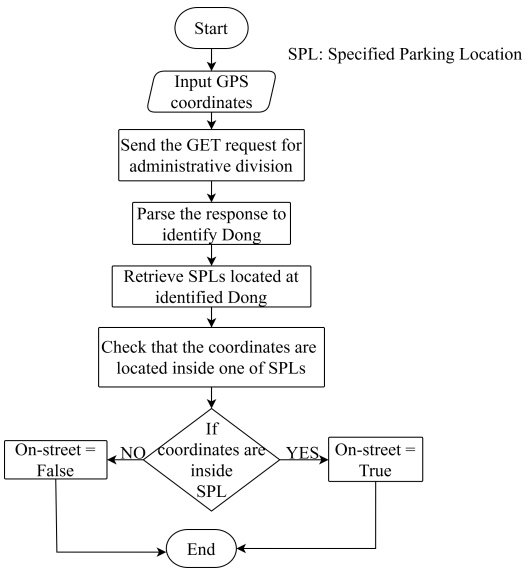


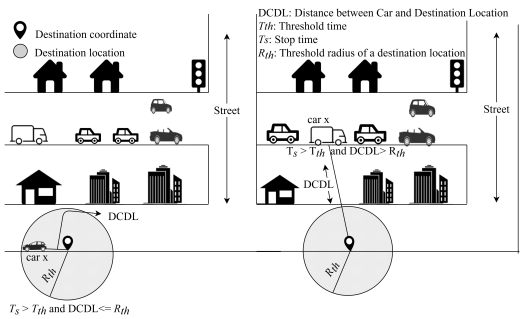
Fig. 11. Validation process of a parking location based on Scheme II.

parked car is located at one of the specified parking locations, and otherwise as off-street. Both the Schemes in the system forward the collected APSI for further validation in the case of on-street parking; otherwise, the APSI is excluded. Additionally, the system with Scheme II stores the APSI along with its 'Dong' address. Figure 11 depicts the validation process of a parking location used in Scheme II.

3.2 Validation of the Movement States of Cars

The DD identifies the movement states of a car, i.e., parked or moving, based on the duration of its specific stopping time, and thus, the DD may erroneously indicate that cars are parked in places like traffic lights, traffic congestion, and temporary stops. The misidentified car causes the uploading of erroneous APSI because the DD collects and uploads APSI whenever the car is in the parked state. Therefore, it is important to validate whether the parked car is in a true parking state or a false parking state, which is done with the API. As shown in Figure 12, the car is truly parked if the car is inside a threshold radius, and the stop time of the car is greater than a threshold time; otherwise, the car is falsely parked.

To verify the parking states of a parked car, the API takes APSI and the threshold radius as input parameters. After receiving the input parameters, the API requests the destination coordinates of the respective car from the database, which is stored in the database whenever the user chooses its



(a) (b)

Fig. 12. Parking states: (a) true parking (b) false parking.

destination in the mobile application. If the destination coordinate of the user is unavailable by default, the parking state of the car is false parking, and thus, additional information is attached to the uploaded ASPI, i.e., parking state is false; otherwise, the distance between the destination coordinate and the current user's coordinate is computed. The computed distance is compared with the threshold radius, and the parking state is set as true if the distance is smaller than the threshold radius; otherwise, the parking state is set as false, as shown in Figure 13.

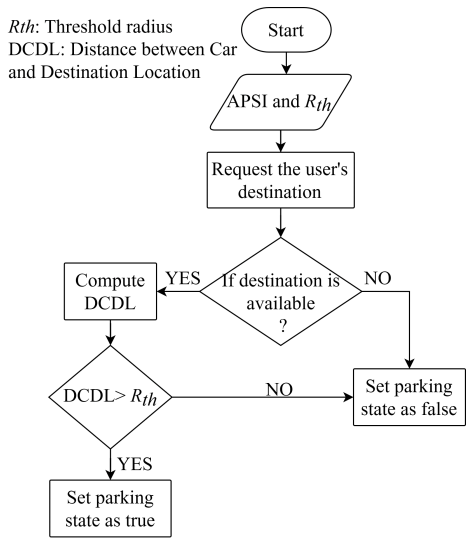


Fig. 13. Validation of the parking states: true or false.

3.3 Selection of Feasible Parking Locations

The main role of the API is an interface between the database and the DD for collecting APSI, which was described in Section III. A and B. Moreover, the API has another role in providing FPLs to a requesting user. For this purpose, firstly, the API takes such parameters as the user's current and destination locations. Secondly, it retrieves each APSI from the database and stores all APSI in a *list* whose coordinates are inside of a predefined threshold radius, R_{th}, using algorithm 1 given below.

When algorithm 1 returns the *list,* algorithm 2 selects the most feasible parking locations from the *list.* To select the FPLs, the API computes the walking distance and driving distance using both *the Pedestrian route guide API* and the *Car route guide API* of *Tmap*, respectively, which are provided by SK Telecom[30]. The walking distance is computed between each parking location and user destination, while the driving distance is computed between the user's current location and the parking locations. Finally, the API provides FPLs to the requesting user.

Our proposed system with Scheme I selects a *list*

Algorithm 1. Selecting all APSI within a threshold radius (R_{th})

1: Select all APSI from the database.
2: While $i < n_1$ where n_1 is the total amount of APSI
3: Compute the distance (d) between the GPS coordinate in the i^{th} APSI and the user destination using the haversine formula
4: If $d < R_{th}$
5: Add the APSI to the *list* having coordinates within the R_{th}
6: i ++
7: Return the *list*

Algorithm 2. Selecting FPLs from the list

1: Take the *list* as an input
2: if $n_2 < 3$ where n_2 is the total amount of APSI in the *list*
3: Select parking facilities near the user's destination using the *Tmap Places API.*
4: else
6: while $j < n_2$
7: Compute the walking distance between the coordinate of the j^{th} APSI in the *list* and the user's destination using the *Pedestrian route guide API.*
8: Sort the APSI in the *list* based on the walking distance.
9: while $k < 3$
10: Compute the driving distance between the coordinate of the k^{th} APSI in the *list* and the user's destination using the *Car route guide API.*
11: Send the first three APSI from the *list* to the requesting user.

from all APSI, which consumes more time with the rising amount of APSI, whereas, with Scheme II, the API separates the collected APSI based on a 'Dong' before storage. Afterward, a *list* is selected from the APSI of cars parked at the 'Dong' of the user's destination location instead of the total amount of APSI. A minor change is done at step 2 of algorithm 1, where n_l becomes the amount of APSI at the 'Dong' of the destination location to minimize the time consumption.

Ⅳ. Field Trials and Performance Evaluation

We developed the proposed system as a prototype and evaluated its performance through field and laboratory trials in terms of 1) the accuracy of parking locations, 2) the average time required for collecting APSI in the server, 3) the average time required for providing FPLs, and 4) the average time required for selecting FPLs.

4.1 System Implementation

We implemented the DD, server, and mobile application, considering the system requirements described in Section II.

4.1.1 Detection Device.

We implemented a prototyped DD using a Raspberry pi, a GPS receiver, an LTE modem, and two ultrasonic sensors, as shown in Figure 14. The Python programming language was used for the Raspberry pi, which served as the controller of the DD. Table 2 shows the device specifications used in the DD.

For our field trials, the system parameters for the DD, such as threshold time, threshold distance, and threshold radius, were set to 5 minutes, 4 meters, and 500 meters, respectively. In addition, the time interval of the GPS coordinates was initialized by 30 seconds to identify 'move' or 'stop' of the car. During the field trials, whenever a car was parked, the DD collected APSI and uploaded it to the server in the system.

Table 2. Device specifications used in the DD.

Device	Specifications
Raspberry pi	Raspberry pi 3, Model B
GPS receiver	NEO-6 series
LTE modem	ALCATEL L800Z
Ultrasonic sensor	HC-SR04

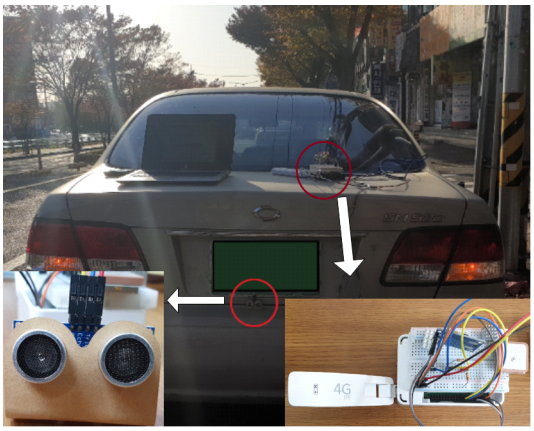


Fig. 14. The DD mounted on the car for field trials.

4.1.2 Server

The server in the proposed system consists of a database and API. We used the MongoDB for the database and the Express framework of the Node.js for the API. Different platforms, namely, MongoDB Atlas[27], and Heroku[28], were used to deploy the API and the database, respectively. Specifically, the MongoDB and Node.js used version 4.2.0 and version 10.16.3, respectively.

On receiving APSI, the server applies the two Schemes I and II to validate the parking location of a car, and the validation result of each Scheme is stored separately in specific databases. The databases were analyzed to determine the accuracy of the parking locations obtained with different schemes. The server also validates parking states of the car regardless of the schemes. With Scheme I and Scheme II, the server provides FPLs to a requesting user but provides the locations of parking facilities in the absence of on-street parking spaces.

4.1.3 Mobile Application

The mobile application was developed by the Android Studio 3.5 and Java. The application uses

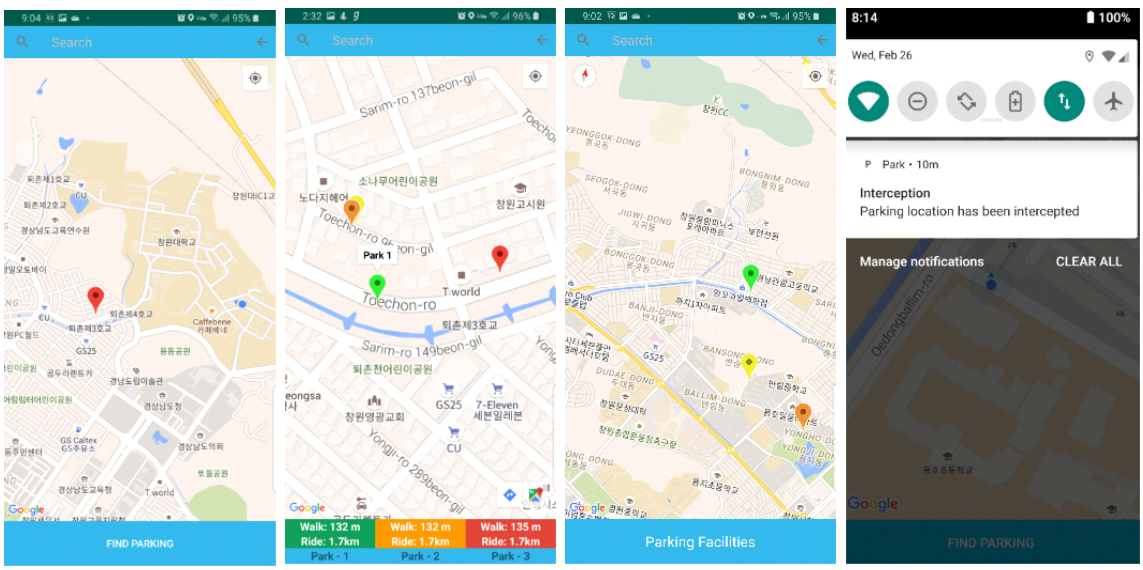
Firebase Cloud Messaging (FCM) to receive interception messages and a NAVER map for car navigation. When a user requests a parking location, we consider three possible cases: available on-street parking, unavailable on-street parking, and an interception. Figure 15 shows the main menu in which a user can select its destination and request FPLs (Figure 15.a). The user receives the FPLs (Figure 15.b) when on-street parking spaces are available near the destination; otherwise, the user receives parking facilities (Figure 15.c). The received FPLs include such details as walking and driving distances from the destination.

A user receives a warning message when other drivers intercept an FPL selected by the user during navigation (Figure 15.d), and then the mobile application automatically suggests other optional FPLs. As a worst-case scenario, when all of the received optional FPLs have already been occupied by other drivers, i.e., *interceptions,* the mobile application requests nearby parking facilities from the server. We created a situation for the interception by blocking an ultrasonic sensor in the car.

4.2 Laboratory and Field Trials

We performed both field and laboratory trials using the development system because it is not realistic to deploy a large number of cars to generate APSI and to create interceptions in the field. Field trials were performed to evaluate the performance measure and the accuracy of the parking locations, considering the system parameter values given by Table 3. In particular, several designated parking locations were stored in the database, and a car with a DD was used to obtain multiple APSI in residential and commercial areas. Both Scheme I and Scheme II were applied to the system to validate the accuracy of the parking locations.

We also performed laboratory trials to evaluate the other performance measures, namely, the average time for collecting APSI, the average time for providing FPLs, and the average time for selecting optional FPLs. In the laboratory, we generated a large amount of APSI to emulate many parking spaces. Table 4 depicts the system parameter values set for the laboratory trials.



(a) (b) (c) (d)

Fig. 15. Mobile application: (a) destination selection, (b) feasible parking locations, (c) parking facilities and (d) interception.

Table 3. Setup values of system parameters for field trials.

System parameters	Setup values
Stop time (T_s)	0 Sec
Threshold distance for movement of a vehicle (D_{mth})	1 m
Threshold time (T_{th})	2 min
Threshold radius (R_{th})	500 m
Threshold distance for the availability of parking (D_{ath})	4 m
Time interval to retrieve GPS coordinate	30 sec
Additional spaces of specified parking locations on each side of the streets.	4 m

Table 4. System parameter values for laboratory trials.

System parameters	Setup values
Threshold time (T_{th})	30 Sec
Threshold distance for the availability of parking (D_{ath})	1 m
Number of APSI	{25, 50, 100, 5000}

4.3 Performance Evaluation and Discussion

Through the laboratory and field trials, we obtained test results to evaluate the performance of the development system.

4.3.1 Accuracy of Parking Locations

It is crucial to identify whether the parked location is an on-street or off-street location because the development system may mistakenly suggest off-street parking to the users requesting for on-street parking. Firstly, we performed field trials to evaluate the *accuracy of parking locations*, which means the system correctly identifies that the parked location is an on-street location when a car is parked at an on-street location:

$$Accuracy\ of\ parking\ locations = \frac{number\ of\ \ parked\ cars\ identified\ correctly}{Total\ number\ of\ parked\ cars}.$$

Out of 15 trials to collect APSI, ten trials were performed in residential areas and the remaining trials in commercial areas. The obtained locations of parking spaces were plotted in Google Maps to get the *number of parked cars identified correctly*. Figure 16 shows that both Schemes I and II identified the parking locations with 60% accuracy in the commercial areas, while with Scheme II, the accuracy of parking locations was improved over Scheme I in residential areas, in which the improvement was 20%.

In particular, with Scheme I in the commercial areas, it is restricted to increase the accuracy of parking locations because high buildings affected the accuracy of the GPS receiver, and thus the received GPS coordinates were far from the actual location, as shown in Figure 17.a. However, with Scheme II, which designated wide boundaries, the accuracy of the GPS receiver was only slightly affected due to lower buildings in the residential areas, and thereby Scheme II achieved higher accuracy (Figure 17.b). Even though the parking locations were identified with higher accuracy in the residential areas, the system could mistakenly detect free parking spaces on private property, which is not legal. This issue remains a future study that may be resolved with big data technologies.

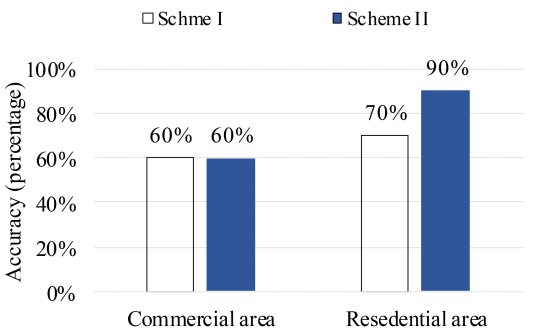


Fig. 16. Accuracy of parking locations in commercial and residential areas.

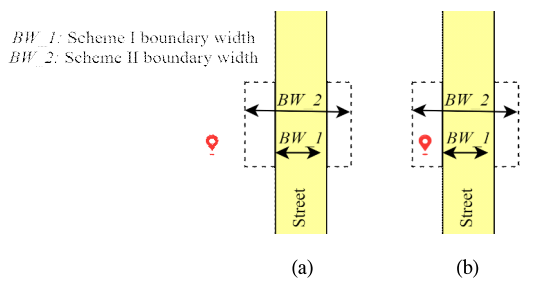


Fig. 17. Effect in the accuracy of GPS receiver: (a) commercial area (b) residential area.

4.3.2 Average Time Required for Collecting APSI

The average time required for collecting APSI means the time for storing APSI and updating APSI. The development system quickly gathers sufficient APSI by storing APSI during a minimum time, which increases the possibility of finding parking locations. These trials were performed with three DDs in the laboratory: a log file that consisted of requests and response times were analyzed to compute the average time for storing APSI and updating APSI. Figure 18 shows that the DDs succeeded in storing APSI within 3.5 seconds and updating the APSI within 2 seconds. This was because the server performs multiple validations before storing the APSI. In particular, the updating time also represents the time required to send a notification message during an interception. This performance was evaluated for both Schemes.

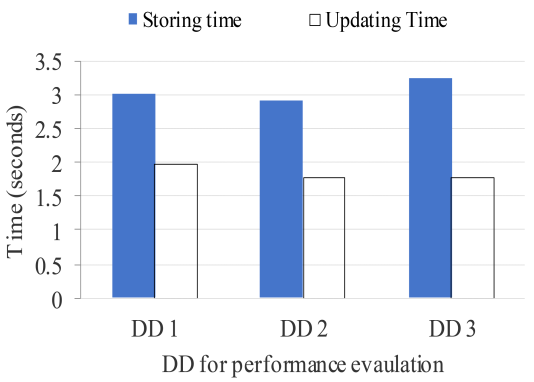


Fig. 18. The average time required for storing APSI and updating APSI.

4.3.3 Average Time Required for Providing FPLs

The required time for providing FPLs means the processing time for APSI, and it also means that the proposed system is capable of providing FPLs considering the amount of APSI. We generated multiple APSI to emulate a large number of parking spaces because it was not realistic to deploy many cars with DDs. In addition, we specified six parking locations at three 'Dongs' in both residential and commercial areas, in which APSI was generated randomly. The processing time of the server was computed using an HTTP profiler[5] plugin of Android Studio, and available parking spaces (the amount of APSI) were generated up to 5000.

Figure 19 shows that the amount of APSI increases, and the difference in the time for providing FPLs also increases. In particular, Scheme II performed slightly faster for 5000 APSI because the APSI located at a specific 'Dong' was processed out of three 'Dongs.' However, Scheme I outperformed Scheme II when there was less APSI because Scheme II spent time identifying the 'Dong' of a user's destination. This result shows that the development system could provide FPLs within 6 seconds despite an enormous amount of APSI.

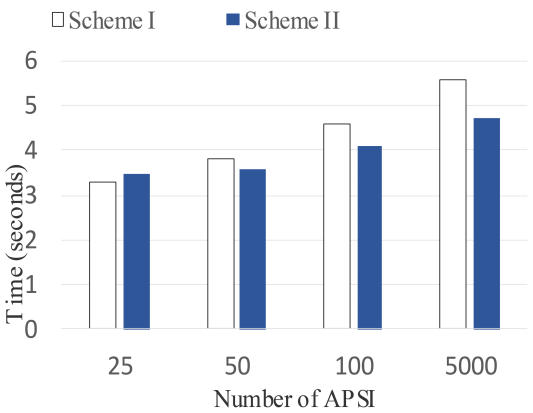


Fig. 19. The average time required for providing FPLs.

4.3.4 Average Time Required for Selecting optional FPLs.

The time required for selecting optional FPLs means the time for a mobile application to select optional FPLs in case of an interception. An interception may happen any time, so the development system provides three FPLs to a user requesting for parking locations. We performed 20 trials at the laboratory, in which the FPLs were randomly assigned to the mobile application, and an interception notification was created by using the Postman[6]. The time for selecting optional FPLs is

5) The profiler collects information on HTTP requests
6) Postman is a tool for interacting with HTTP APIs i.e. constructing requests and reading responses.

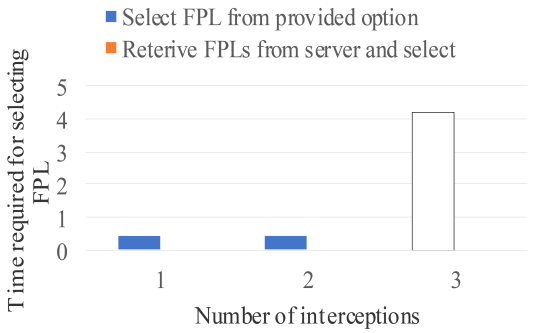


Fig. 20. Average time required for selecting FPLs.

stored in a log file. Figure 20 depicts that until two interceptions, the mobile application selects the optional FPLs instantly; However, the third occurrence of an interception increases the time since the application requests the server for FPLs. Note that this performance measure has no relationship with both Schemes.

V. Conclusion and Further Work

The increase in vehicles in urban areas often creates a shortage of parking spaces, which consequently produces undesirable issues such as waste of time, air pollution, and traffic congestion. We proposed a connected car-based parking location guidance system that consists of a detecting device for available parking spaces, a cloud server, and a mobile application for drivers. Our proposed system provides on-street parking space information to requesting drivers through an LTE cellular network. Furthermore, the system provides parking facilities in the absence of on-street parking spaces and handles an interception of parking location from other drivers by detecting and updating a specific available parking space in real time.

With the proposed system, unnecessary movements to intercepted parking locations could be significantly reduced. The proposed system was implemented as a prototype and evaluated its performance through field trials in residential and commercial areas. The evaluation results confirmed that our proposed system swiftly collects, updates, and provides available parking spaces with higher accuracy to vehicle drivers, thereby increasing the user's possibility of finding parking spaces. Applying big data technology and higher quality GPS receivers improves the performance of the system for determining legal parking locations, real-time occupancies of parking facilities, and the accuracy of GPS coordinates, which remains as our future study.

References

[1] K. Ashton, "That 'Internet of Things' Thing in the real world, things matter more than ideas," *RFID J.*, Jun. 2009, Retrieved Nov. 27, 2019, from http://www.rfidjournal.com/article/print/4986.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

[3] S. Lucero, IoT platforms: enabling the Internet of Things, *IHS Technol.*, 2016, Retrieved Jan. 05, 2020, from https://cdn.ihs.com/www/pdf/enabling-IOT.pdf.

[4] S. K. Datta, R. P. F. Da Costa, J. Harri, and C. Bonnet, "Integrating connected vehicles in Internet of Things ecosystems: Challenges and solutions," in *Proc. 2016 IEEE 17th Int. Symp. WoWMoM*, pp. 1-6, 2016.

[5] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289-299, Aug. 2014.

[6] Telefonica, *Connected Car Industry Report*, 2014. Retrieved Jan. 05, 2020, from https://www.business-solutions.telefonica.com/scripts/vendor/pdf.js/web/viewer.html?file=/media/1466/3-en-connected-car-industry-report-2014.pdf.

[7] B. Danne and P. Hofer, "Connected car business models: State of the art and practical opportunities," *AutoScout24*, pp. 3-20, 2014.

[8] J.-H. Shin and H.-B. Jun, "A study on smart parking guidance algorithm," *Transp. Res.*

Part C Emerg. Technol., vol. 44, pp. 299-317, Jul. 2014.

[9] D. Shoup, "Cruising for parking," *ACCESS Mag.*, vol. 30, pp. 16-22, 2007.

[10] S. Bharadwaj, S. Ballare, Rohit, and M. K. Chandel, "Impact of congestion on greenhouse gas emissions for road transport in Mumbai metropolitan region," *Transp. Res. Procedia*, vol. 25, pp. 3538-3551, 2017.

[11] C. Roman, R. Liao, P. Ball, S. Ou, and M. De Heaver, "Detecting on-street parking spaces in smart cities: Performance evaluation of fixed and mobile sensing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2234-2245, 2018.

[12] H. Wang and W. He, "A reservation-based smart parking system," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, pp. 690-695, 2011.

[13] P. Chippalkatti, G. Kadam, and V. Ichake, "I-SPARK: IoT based smart parking system," in *Proc. ICACCT*, pp. 473-477, 2018.

[14] C. Lee, Y. Han, S. Jeon, D. Seo, and I. Jung, "Smart parking system using ultrasonic sensor and bluetooth communication in internet of things," *KIISE Trans. Comput. Pract.*, vol. 22, no. 6, pp. 268-277, 2016.

[15] R. Yusnita, N. Fariza, and B. Norazwinawati, "Intelligent parking space detection system based on image processing," *Int. J. Innov. Manag. Technol.*, vol. 3, no. 3, pp. 232-235, 2012.

[16] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, pp. 1212-1217, 2016.

[17] Z. Suryady, G. R. Sinniah, S. Haseeb, M. T. Siddique, and M. F. M. Ezani, "Rapid development of smart parking system with cloud-based platforms," in *Proc. 5th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M)*, pp. 1-6, 2014.

[18] SFpark, *SFPark—A Project of San Francisco Municipal Transportation Authority* (SFMTA), 2014, Retrieved Jan. 9, 2020, from http://sfpark.org/.

[19] Worldsensing, *FASTPRK: The Easiest Way to Park*, 2015, Retrieved Jan. 9, 2020, from https://www.worldsensing.com/product/fastprk/.

[20] V. O. Ethos, *Case Study: Guildford*, 2015, Retrieved Jan. 9, 2020, from https://geomii.co.uk/case-study-1-guildford/.

[21] S. Mathur, et al., "ParkNet: Drive-by sensing of road-side parking statistics," *MobiSys'10 - Proc. 8th Int. Conf. Mob. Syst. Appl. Serv.*, pp. 123-136, 2010.

[22] K. Gkolias and E. I. Vlahogianni, "Convolutional neural networks for on-street parking space detection in urban networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4318-4327, 2018.

[23] Y. KC and C. S. Kang, "A connected car-based parking location service system," in *Proc. 2019 IEEE IoTaIS*, pp. 167-171, 2019.

[24] E. Winarno, W. Hadikurniawati, and R. N. Rosso, "Location based service for presence system using haversine method," in *Proc. 2017 ICITech*, pp. 1-4, 2017.

[25] JSON, *Introducing JSON*, Retrieved Jan. 9, 2020, from https://www.json.org/json-en.html.

[26] Mozilla, *POST - HTTP*, Retrieved Feb. 24, 2020, from https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST.

[27] MongoDB Inc, *Database-as-a-Service*, Retrieved Dec. 23, 2019, from https://www.mongodb.com/cloud/atlas.

[28] Heroku, *Cloud Application Platform*, Retrieved Dec. 23, 2019, from https://www.heroku.com/home.

[29] Naver, *Naver map Reverse Geocoding API documentation*, Retrieved Dec. 23, 2019, from https://apidocs.ncloud.com/en/ai-naver/maps_reverse_geocoding.

[30] SK Telecom, *API documentation*, Retrieved Dec. 23, 2019, from https://openapi.sk.com/resource/apidoc/indexView.

Yugesh KC

He received his B.S. degree in Computer Science and Information Technology from Tribhuvan University, Nepal, in 2016 and currently pursuing (2018-2020) his M.S. degree in eco-friendly offshore plant FEED engineering at Changwon National University, Changwon, Republic of Korea. He joined the Shree Krishna Engineering Associate, Kathmandu, Nepal in June 2016, where he worked as a software developer until July 2018. His research interests are in the areas of the Internet of Things (IoT), mobile application development, and wireless communication networks.

[ORCID:0000-0002-0998-6457]

Chang Soon Kang

He received his B.S. degree from Kyungpook National University in 1984, M.S. degree from Yonsei University, Seoul, Korea, in 1986, all in electronics engineering, and Ph.D. degree in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology in 2001. He joined the Electronics and Telecommunications Research Institute (ETRI), Taejon, in October 1989, where he was involved in several projects. From 1989 to 1996, he was engaged in the development of the IS-95 based CDMA Cellular System. From 1997 to 1998, he worked on a CDMA based wireless local loop system. He was also involved in developing the WCDMA based IMT-2000 system and Wireless Broadband Internet system (WiBro) from January 1999 to February 2003. In 2003, he joined a faculty member of the Changwon National University, where he is a full Professor in the Dept. of Information & Communication Eng. From September 2012 to August 2013, he was with the Department of Computer Science and Telecommunications (DISI) of the University of Trento, Trento, Italy, as a Visiting Scholar. His research interests are in the areas of wireless communications and networks with typical emphasis on Heterogeneous Cellular Network (HetNet), Internet of Things (IoT), and Vehicle to Everything (V2X).

[ORCID:0000-0001-9979-4389]