J Inf Process Syst, Vol.13, No.4, pp.951~969, August 2017
https://doi.org/10.3745/JIPS.03.0078

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Traffic Information Service Model Considering Personal Driving Trajectories

Homin Han* and Soyoung Park*

Abstract

In this paper, we newly propose a traffic information service model that collects traffic information sensed by an individual vehicle in real time by using a smart device, and which enables drivers to share traffic information on all roads in real time using an application installed on a smart device. In particular, when the driver requests traffic information for a specific area, the proposed driver-personalized service model provides him/her with traffic information on the driving directions in advance by predicting the driving directions of the vehicle based on the learning of the driving records of each driver. To do this, we propose a traffic information management model to process and manage in real time a large amount of online-generated traffic information and traffic information requests generated by each vehicle. We also propose a road node-based indexing technique to efficiently store and manage location-based traffic information provided by each vehicle. Finally, we propose a driving learning and prediction model based on the hidden Markov model to predict the driving directions of each driver based on the driver's driving records. We analyze the traffic information processing performance of the proposed model and the accuracy of the driving prediction model using traffic information collected from actual driving vehicles for the entire area of Seoul, as well as driving records and experimental data.

Keywords

GPS-to-Road Mapping Strategy, Personal Trajectory, Traffic Information System, Trajectory Estimation

1. Introduction

With the widespread use of IoT technology, there has been active research on connected cars that can build a safe and intelligent transportation system (ITS) through real-time information communication between vehicles. However, various service models that can realize an early connected car environment by combining existing vehicles and smart mobile devices have been developed because the supply of smart cars with inter-vehicle communication and computing functions remains insufficient, and it is too early stage to establish a connected-car environment.

Recently, a lot of driver assistant mobile applications that provide useful information to the driver such as a car navigation application, a driving pattern analysis application, and a vehicle condition check application have been developed. The development of services that can implement the existing ordinary cars as smart cars in combination with smart mobile devices that provide with portability,

Manuscript received April 18, 2017; first revision May 17, 2017; accepted May 18, 2017.
Corresponding Author: Soyoung Park (soyoungpark@konkuk.ac.kr)
* Dept. of Software, Konkuk University, Seoul, Korea (hi_test@naver.com, soyoungpark@konkuk.ac.kr)

location information, computing and storage functions, and communication functions is expanding, and the service users are also increasing.

The most essential information that a driver needs while driving is the real-time traffic information for his or her driving route. Through the combination of car and smart mobile device, it is possible to obtain traffic information from each vehicle in real time, share the traffic information between vehicles, and it is also possible for drivers to obtain traffic information for a desired area in real time. Particularly, if the driver can grasp the real-time traffic situation in advance to the route to be moved forward, it is possible to more efficiently travel by responding to the current traffic situation more quickly.

In this paper, we propose a practical traffic information service model that collects traffic information in real time through a smart device application mounted in each vehicle, and which obtains the latest traffic information on the location desired by the driver. In particular, we newly propose a personalized traffic information service model that provides the latest traffic information on the most likely driving directions to reach the location requested by the driver. The proposed model considers the moving directions of the vehicle. Currently, real-time traffic information is collected using CCTVs or traffic-collection devices installed in a specific location, but this has the disadvantage of not being able to share traffic information in real time for roads other than specific areas.

In the proposed model, the vehicle generates location information and driving information of the vehicle by utilizing a GPS-receiving function of a smart device and other sensors, and it periodically provides the driving information to a traffic information management server through a vehicle black box application. The traffic information management server can collect image-based traffic information from each vehicle in real time. In addition, the driver can also share in real time online traffic information on all roads provided by all vehicles using the proposed application. In order to practically implement and utilize the proposed model, in this study, we propose specific techniques that focus on the following three topics. (1) We design a traffic data management model that can effectively manage and provide traffic information provided by each vehicle in real time. Second, in order for the traffic information management server to efficiently store and retrieve traffic information, the traffic information should be indexed by an actual road location (or node). (2) We propose a new method to efficiently search for road nodes mapped to GPS positions. Then, when a driver requests the latest traffic information for a specific location from the traffic information management server, the latest traffic information on the most likely driving directions to reach the target position is first provided by considering the driving directions of the vehicle. (3) We propose a prediction model of the driving directions of the vehicle through hidden Markov model (HMM) [1] learning of the driving records of each vehicle.

Finally, we analyze the performance of real-time traffic information collection and processing based on the road information in Seoul. We also analyze the accuracy of the proposed driving probability model by utilizing actual driving records in order to evaluate the performance of the proposed model.

The structure of this paper is as follows. In Section 2, we discuss related works. In Section 3, we explain the components and assumptions of the proposed model. In Section 4, we describe in detail the main functions of the vehicle black box application for the smart device and the search technique for road node search from GPS coordinates as well as the probability model for vehicle driving prediction. In Section 5, we present the traffic information management server, which can efficiently collect and manage traffic information. Then, in Section 6, we analyze the performance of the proposed model, and in Section 7, we conclude this paper.

2. Related Works

In this paper, we present techniques that provide traffic information suitable to a driver by collecting traffic information sensed by each vehicle, analyzing the driving route of the vehicle, and predicting the driving directions of the vehicle. First, Mohan et al. [2] proposed a method of sensing road condition information using various smartphone sensors. In SEER [3], a model was proposed to detect traffic conditions in Shanghai using information sensed by taxis. The technique proposed in [4] also utilizes the information provided by each vehicle to sense the traffic situation in an urban area.

GreenGPS [5] proposed a model that provides the driver with useful personalized services by utilizing traffic information collected through smart devices, and proposed a method to improve the fuel efficiency by analyzing the fuel use status according to each road on which a vehicle is driven. RoadAware [6] proposed a model that provides the driver with road information by analyzing the waiting time of the vehicle at each traffic light on commuter routes, the distance between traffic lights, the time taken for the signal to change, and the traffic volume between traffic lights. In [7], the authors proposed a method of searching for an eco-friendly route that is suitable for the driver by considering the driver's driving style and traffic conditions.

Guttman [8] proposed the R-tree as a search technique to search for the road nodes that are closest to GPS coordinates, which is often used to solve the k-nearest neighbor (k-NN) problem. The R*-tree [9] was proposed to complement the overlapping of clustering areas owing to the nature of the R-tree, but it does not completely resolve the overlapping problem. A kDB-tree [10] in which the overlapping problem does not occur was proposed, but there may be a region that contains only a very small number of nodes owing to the partition rule of the kDB-tree itself. Recently, the GB-tree [11] using the GeoHash was proposed. While it is not a clustering method, it is used to calculate and store a high-precision GeoHash for a specific space. This method is similar to the proposed method, yet the performance of the proposed method is better than that of the GB-tree because the size of the tree is relatively large, and additional operations are required for the search-tree generation.

With regard to driving-route prediction, T-Drive [12] proposed a model that provides a driver with smart driving directions based on the driving records of taxi drivers, who are always searching for the shortest route. This is similar to the proposed study in terms of predicting the driving directions based on the driving records of the vehicle, but it is different in that our study provides general driving directions for all vehicles.

3. System Model and Assumptions

The proposed traffic information service model consists mainly of vehicles, smart mobile devices equipped with a vehicle blackbox application, and the traffic data management system (TDMS).

We assumed that a smart mobile device is equipped with a GPS receiver, camera, computational capabilities, data storage, and wireless communication functions.

Vehicular Blackbox Application (hereinafter referred to as TrafficAPP) provides a road map, a blackbox function, a function to store and learn driving records, a mapping function of actual road information corresponding to GPS coordinates, and a driving information generation and transmission function. In addition, it receives a user request and provides the user with a request result by

communicating with the traffic information management server.

The TDMS stores and manages in real time driving information of various areas provided by each vehicle, processes the user's traffic information request for a specific location, and provides the latest information on the location.

In summary, the TDMS collects traffic information in real time from vehicles that use the TrafficAPP, and the vehicle can request traffic information for the desired location using the TrafficAPP from the TDMS, and the TDMS provides the latest traffic information on the location, as illustrated in Fig. 1.

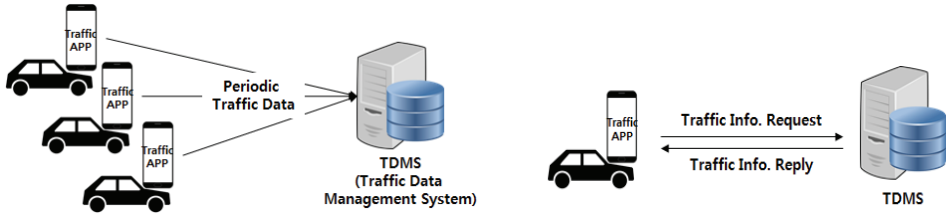


Fig. 1. Communication between vehicles and traffic data management system (TDMS).

Road information uses the road node information provided by the Ministry of Land, Infrastructure, and Transport. A road node represents the points at which two or more roads intersect, merge, or separate, and the section between nodes is represented by a link. Each node has its own unique ID. In this study, virtual road nodes are also created and utilized along with real road nodes to realize a more accurate representation of road information. The length of the link between nodes differs depending on the road type (public road or highway) and the area (urban area or suburban area). Therefore, in the case where the length of the link exceeds the threshold determined by the system, the link is divided by the threshold unit, and a virtual node is also created at the threshold point. For a curved road, a virtual node is added to the portion corresponding to the vertex of the curve (provided by the Ministry of Land, Infrastructure, and Transport). The ID of the added node is generated by adding the lower two digits to the ID of the node with the smaller ID among the adjacent nodes connected to the link at which the additional node is positioned. Each time a node is added, the additional lower digit incremented by 1. It is possible to identify which virtual node is derived from which node as the virtual nodes have the same ID prefix value as the adjacent real nodes.

We assumed that the proposed TrafficAPP and the TDMS server have road-node network information (hereinafter referred to as RNetwork) that indicates the adjacent state between all of the nodes, including the virtual nodes. Because the positions of the road nodes and the distances between nodes are fixed, the shortest distance and shortest path between all nodes can be determined based on the Euclidean distance. We assumed that the information about these is available. All notations used in the paper are summarized in the following Table 1.

4. TrafficAPP: A Vehicular Blackbox Application for Smart Mobile Devices

This section describes the TrafficApp in detail. First, we describe the main functions of TrafficAPP, and we describe GPS coordinates and a proposed road-node matching technique to collect efficient

traffic information. In addition, we describe in detail a method to build a probability model for the driving directions of the vehicle in order to provide traffic information for each vehicle.

Table 1. Notations

Notation	Description
TDMS	Traffic data management system
TrafficAPP	A mobile application for a car
TrafficMSG	A text-based traffic message periodically generated by TrafficAPP
TrafficIMG	A image-based traffic message periodically generated by TrafficAPP
TrafficRQST	A traffic request message generated by TrafficAPP
DPP	Data processing pipeline for TDMS
MsgSV	A storage for TrafficMSG
ImgSV	A storage for TrafficIMG
RNetwork	A road node network
CNID	The ID of node that is the closest to given GPS
PNID	The ID of node that a car previously passed by
Speed	The current speed of a car
Timestamp	The current time
HS-Tree	Hash search tree to find a node id corresponding to given GPS
n	The total number of road nodes
N_i	The i-th node in RNetwork
$\|N_i\|$	The # of nodes adjacent to N_i
$N_{i,k}$	The k-th adjacent node to N_i
TJ	A series of node ids that a car passed by sequentially
TJ_i	The i-th visiting node in TJ
O	A series of observed events that a car made at each node
O_i	The event happened at TJ_i
A_{ij}	The transition probability from N_i to N_j
$B_i(k)$	The probability that an event k occurs at N_i
$I(i)$	The probability that a trajectory begins at N_i
$TGPS$	The GPS coordinates of a target position
N_T	The node closest to $TGPS$
N_{TPrev}	The final visiting node before reaching $TGPS$

4.1 Functions of TrafficAPP

TrafficAPP consists of a black box shooting and storage module, a driving record analysis module, and a real-time traffic information management module, as shown in Fig. 2. It uses the camera and GPS functions of the smart device. To utilize the road information, a hash search tree that is capable of searching for road nodes matched with electronic maps such as Google Map, road-node networks, and GPS coordinates is given. The blackbox module captures current driving situations ahead of the vehicle using the camera function of the smart device. The captured image is divided into a predetermined size unit, and it is stored and managed as a file. The images that are captured periodically according to user settings are utilized for the generation of real-time traffic information. The driving record analysis module learns the driving route each time the vehicle moves drives. It then updates the probability

model of the moving route for each road node. The real-time traffic information management module performs the function of providing driving information to the traffic information server by generating driving information periodically, and it also performs the function of requesting real-time traffic information for the driver's desired location.

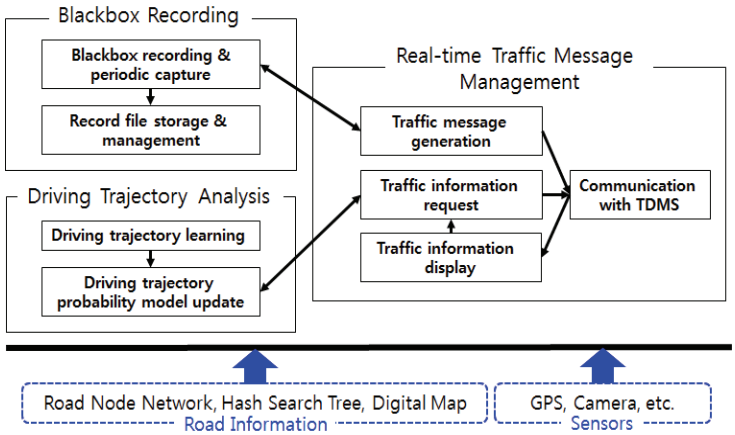


Fig. 2. System configuration of TrafficAPP.

4.2 Traffic Message

Each vehicle transmits a traffic message periodically to the TDMS via the TrafficAPP. The traffic message is provided as a text message (hereinafter referred to as TrafficMSG) that indicates the current driving state and a static image frame (hereinafter referred to as TrafficIMG) that is periodically extracted from the images recorded using the TrafficApp blackbox function. However, the TrafficIMG is optionally provided only if the driver has activated the TrafficAPP blackbox function. The transmission period of the TrafficMSG can be selectively set by a certain time unit (e.g., a 1-s unit) or a certain driving distance (e.g., a 10-m unit). The TrafficIMG is transmitted when at a certain driving distance by a certain driving distance. The basic message formats of TrafficMSG and TrafficIMG are as follows.

TrafficMSG = {"TMG" || GPS-coordinates || CNID || PNID || Speed || Timestamp};

TrafficIMG = {"IMG" || GPS-coordinates || CNID || PNID || Speed || Timestamp} + Captured Image File;

The CNID indicates a road node ID mapping to the GPS coordinates contained in the current TrafficMSG. The PNID is a node ID that a vehicle previously passed, and it indicates the CNID value of the immediately preceding TrafficMSG. The method used to find a road node that maps to GPS coordinates is explained in detail in the next section.

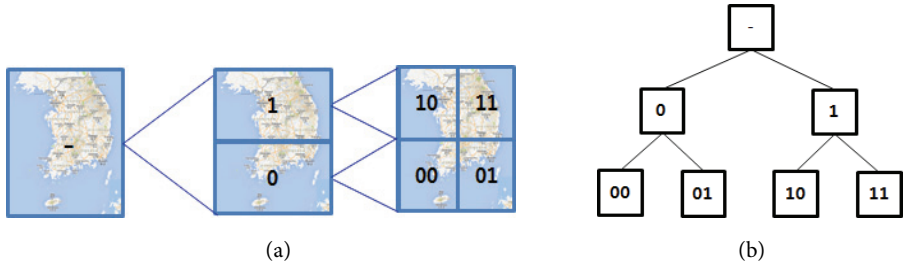
4.3 GPS Position to Road Node Mapping Strategy

In this section, we describe the proposed hash search tree (HS-tree) generation method to search for a road node closest to a given GPS position. First, the whole area including the entire road is divided into

a grid of the same size in a manner similar to the GeoHash. A hash code is assigned to the final divided space. Let the final divided space be a terminal. All of the GPS coordinates contained in one terminal are mapped to a unique hash code, and the road nodes are clustered on a terminal unit basis. At this time, in the case where the number of road nodes included in each terminal is very small, the merging process is repeated so that the number of nodes included in all the terminals is similar by merging with the adjacent terminal. Here, the hash value of the merged terminal is changed through merging. As the final outcome, a hash search tree is constructed to efficiently search for the final merged terminal and hash code. The proposed scheme consists mainly of four steps: (1) hash code assignment by space partitioning, (2) merger of terminals, (3) hash search tree construction, and (4) road node mapping. Details of each step are as follows:

Hash Code Assignment

Let the whole area including the entire road be the initial two-dimensional (2D) space. We partition the initial space by the same size vertically and horizontally in sequence. Then, we repeat the same process until the final divided space is reduced to a predefined space. Each time the space is partitioned, we append 0 and 1 to the existing code values in each partitioned space, as shown in Fig. 3(a). For each iteration of the space partition, a binary search tree can be created for the added code value, as shown in Fig. 3(b). The end node of the generated binary search tree is called a terminal, and the binary code assigned to each terminal is a hash code for that terminal.



(a) (b)

Fig. 3. Space partitioning and hash-code assignment. (a) Space partitioning, (b) initial binary search tree.

Merger of Terminals

Each terminal node of the generated binary search tree includes ID information of all road nodes in the corresponding terminal space. Here, the number of road nodes included in the terminal varies depending on the local characteristics of each terminal. For a terminal located in a city area, the number of road nodes is relatively large, while the corresponding number is small or zero in mountainous or coastal regions. To match the number of nodes similarly included in each terminal, a terminal for which the number of road nodes included in the terminal is less than the minimum threshold set by the system is merged with an adjacent terminal. A merger is performed only if both of the following conditions are met: (1) the length of the hash codes of the two terminals is the same, and there is only a 1-bit difference in the remaining bit strings, excluding the common prefix. (2) The number of road nodes included in the merged terminal does not exceed the maximum threshold set by the system.

When two terminals merge, the hash code for the merged terminal is changed to 0, with the

exception of the hash code that is common to both terminals prior to merging. The terminal merging process is shown in Fig. 4.

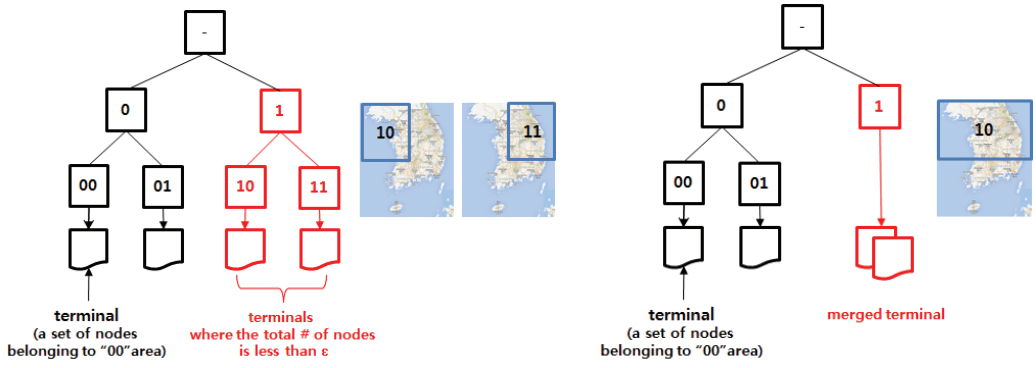


Fig. 4. Merger of terminals. (a) Initial terminals before merging, (b) after merging of terminals.

HS-Tree: Hash Search Tree Construction

When terminals merge, the terminal level on the initial binary search tree changes, as shown in Fig. 4. To make the search times for all terminals the same, it is converted into a B+ tree with the hash codes for merged terminals as keys. The finally converted B+ tree is called a hash search tree. As a result, there are terminals at the end of the final hash search tree. The terminals with the same hash prefix have the same parent node.

Node Mapping Strategy

Given the GPS coordinates, we describe how to determine the hash code and the road node through the hash search tree. The hash code corresponding to given GPS coordinates is determined by the following algorithm.

```
// COORD : GPS coordinates
// MIN_COORD : Min coordinates of the initial region
// MAX_COORD : Max coordinates of the initial region
HashGPS (COORD) {
   HASH = 0;
   For 1 to ACCURACY
        HASH << 1; //Left shift
        MID_COORD = (MIN_COORD + MAX_COORD) / 2;
        If COORD is bigger than MID_COORD
               HASH += 1;
               MIN_COORD = MID_COORD;
        else
               MAX_COORD = MID_COORD;
}
```

For example, when the GPS coordinates (37.54268, 127.07683) for the range of the latitude (ranging from 33 to 39) and longitude (ranging from 124.5 to 132) of the initial space is given, hash code 10110001 is assigned. The process of converting to the hash code is shown in Table 2.

Table 2. Process of converting to the hash code

Step	GPS	Min	Mid	Max	Hash
1	Latitude	33 (initial min)	36	39 (initial max)	1
	Longitude	124.5 (initial min)	128.25	132 (initial max)	0
2	Latitude	36	37.5	39	1
	Longitude	124.5	126.375	128.25	1
3	Latitude	37.5	38.25	39	0
	Longitude	126.375	127.3125	128.25	0
4	Latitude	37.5	37.875	38.25	0
	Longitude	126.375	126.8438	127.3125	1

Once the hash code is determined, the terminal containing GPS coordinates can be found using the hash search tree. Because the terminal at the end of the hash search tree has a merged hash code, there may not be any terminal that exactly matches the given hash code. In this case, we selected a terminal for which the upper prefix value is most matched among terminals at the end of the hash tree. Once the terminal is determined, the GPS information of the nodes included in the corresponding terminal is sequentially compared to determine the closest node ID.

4.4 HMM-Based Driving Trajectory Estimation Model

The TrafficApp continuously learns the driver's driving records, constructs a probability model for the driving route, and uses it to predict the driving directions. Most of the roads are bidirectional, and there are at least four different driving directions at each intersection. Furthermore, the traffic situation may be completely different depending on the driving directions, even for the same location. Therefore, in order for the traffic information to be meaningful to the driver, it should inform the driver about driving directions in advance by predicting the driving route with the highest probability of reaching the target position from the current position of the driver. Therefore, in this section, we describe in detail a method that can be used to build a probability model to predict the driver's driving directions using the driver's driving records, and we then propose a practical prediction algorithm.

4.4.1 HMM-based probabilistic model for driving directions

The TrafficApp periodically generates the TrafficMSG. On each occasion, the node ID at which the vehicle has moved is determined. Let *TJ* be the driving trajectory of the vehicle from start to finish. *TJ* represents the set of a series of consecutive node IDs that the vehicle passed while driving. Here, consecutively duplicated IDs are replaced with one ID value. The proposed driving probability model updates the probability of the vehicle moving between road nodes using HMM learning for new driving trajectories.

Let N_i be the ith node. $|N_i|$ denotes the number of nodes adjacent to N_i, and $N_{i,k}$ denotes the kth adjacent node to N_i. The event that the vehicle can perform at each node is to move to a specific adjacent node. That is, the event value generated at each node is defined as the index value (k) of the

adjacent node into which the vehicle moved from the corresponding node. Because the number of adjacent nodes is different for each node, the number of events that can occur at each node is the same as the number of adjacent nodes. The maximum value (m) that an event can have for the entire road is argmax($|N_i|$).

For the driving trajectory TJ, let the event set observed at each node be O. Let TJ_i be the ith visited node. O_i denotes an event that occurs to move from TJ_i to TJ_{i+1}, and it is an index value that corresponds to TJ_{i+1} among adjacent nodes to TJ_i. At each node, HMM learning is carried out based on the Baum-Welch algorithm once the observation vector O for the moving direction is given at each node. The driving probability model (Θ) consists of (1) the transition probability of a vehicle moving between nodes for all road nodes, and (2) the probability of an event moving in a specific direction at each node. Let A_{ij} be the transition probability that a vehicle will move from N_i to N_j, and let $B_i(k)$ be the probability that an event moving from N_i to $N_{i,k}$ will occur. The initial values of A_{ij} and $B_i(k)$ are determined by Eqs. (1) and (2). $I(i)$, which is the probability ($Prob(TJ_1 = N_i)$) that driving starts at N_i, is initialized to zero.

$$A_{ij} = \frac{1}{n}, \qquad 1 \leq i, j \leq n. \tag{1}$$

$$B_i(k) = \begin{cases} \frac{1}{|N_i|} & \text{if } 1 \leq k \leq |N_i|; \\ 0 & otherwise. \end{cases} \tag{2}$$

At each instant in time, O is given the value of $NewA_{ij}$, and $NewB_i(k)$ is updated as follows. T is equal to $|O|$, and O_t indicates the tth observed event value. Under the condition where TJ and O are given, the value of $I(i)$ is updated first as follows.

$$I(i) = I(i) + 1 \quad \text{if } N_i = TJ_1; \text{ and}$$

$$I(i) = \frac{I(i)}{\sum_{j=1}^{n} I(j)}, 1 \leq i \leq n. \tag{3}$$

In order to obtain the transition probability of moving from N_i to N_j, we need to obtain (1) $\gamma_t(i)$, which is the probability of being at node N_i at any time t, and (2) $\lambda_t(i,j)$, which is the probability of being at node N_i at any time t and being at node N_j at any time t+1. First, $\gamma_t(i)$ which is the probability of being at node N_i at any time t regardless of the state that it is in time (except for t), is determined by $\alpha_t(i)$, which is the probability of being at node N_i at time t and the occurrence of $O_1,\ldots,O_t$ until t and $\beta_t(i)$, which is the probability of occurrence of $O_{t+1},\ldots,O_T$ from t+1 under the condition of being at node N_i at time t. Both $\alpha_t(i)$ and $\beta_t(i)$ are determined by the following Eqs. (4) and (5), respectively.

$$\alpha_t(i) = Prob(O_1, O_2, \ldots, O_t, N_i \mid \Theta) = \left[\sum_{j=1}^{n} \alpha_{t-1}(j) A_{ji}\right] \cdot B_i(O_t), \quad 1 \leq i \leq n,$$
$$\text{where } \alpha_1(i) = I(i), \quad 1 \leq i \leq n. \tag{4}$$

$$\beta_t(i) = Prob(O_{t+1}, O_{t+2}, \ldots, O_T \mid N_i, \Theta) = \left[\sum_{j=1}^{n} A_{ij} B_j(O_{t+1}) \beta_{t+1}(j)\right], 1 \leq i \leq n, 1 \leq \mathrm{t} \leq \mathrm{T} - 1,$$
$$\text{where } \beta_T(i) = 1, \ 1 \leq i \leq n. \tag{5}$$

Therefore, $\gamma_t(i)$ is determined by Eq. (6).

$$\gamma_t(i) = Prob(N_i|O,\Theta) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{n}\alpha_t(j)\beta_t(j)}, \quad 1 \le t \le T, 1 \le i \le n. \tag{6}$$

$\lambda_t(i,j)$ is the probability of a transition from N_i to N_j at any time t, and indicates the probability of $\alpha_t(i)A_{ij}B_j(O_{t+1})\beta_{t+1}(j)$, which is determined by Eq. (7) below.

$$\lambda_t(i,j) = \frac{\alpha_t(i)A_{ij}B_j(O_{t+1})\beta_{t+1}(j)}{\sum_{x=1}^{n}\sum_{y=1}^{n}\alpha_t(x)A_{xy}B_y(O_{t+1})\beta_{t+1}(y)}, 1 \le t \le T-1, 1 \le i,j \le n. \tag{7}$$

From the result, $NewA_{ij}$ is equal to the probability of transition from N_i to N_j divided by the probability of transition to N_i. It is represented by Eq. (8) as follows.

$$NewA_{ij} = \frac{\sum_{t=1}^{T-1}\lambda_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}, 1 \le i,j \le n \tag{8}$$

$NewB_i(k)$ is equal to the probability of occurrence of an event moving from N_i to $N_{i,k}$ divided by the probability of being at N_i. It is represented as follows.

$$NewB_i(k) = \frac{\sum_{t=1 \text{ s.t. } o_t=k}^{T}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)}, 1 \le i \le n, 1 \le k \le |N_i| \tag{9}$$

Finally, A_{ij} and $B_i(k)$ are updated as shown in Eq. (10), and reflects learning results for $NewA_{ij}$ and $NewB_i(k)$, respectively. Here, weight (ω) can be applied to $NewA_{ij}$ and $NewB_i(k)$ to adjust the reflection rate of the learning results. The weight has a value between 0 and 1. When the weight is 1, the transition probability is completely changed to the newly learned $NewA_{ij}$ and $NewB_i(k)$ values.

$$\begin{aligned} A_{ij} &= (1-\omega)A_{ij} + \omega NewA_{ij}; \\ B_i(k) &= (1-\omega)B_i(k) + \omega NewB_i(k); \end{aligned} \tag{10}$$

Whenever a new driving trajectory occurs, the above learning process is repeated to continuously update the transition probability for each node and the event occurrence probability at each node.

4.4.2 Driving trajectory estimation strategy

Based on the driving probability model described in the previous section, the TrafficApp determines the possible directions by predicting the driving route from the current position to the target position when the driver requests real-time information about a specific area using the TrafficApp. Then, it requests traffic information about the driving directions from the TDMS. The basic strategy is to select as the final visiting node the node with the highest arrival probability of the nodes that are closest to the target position as the final visiting node and request the traffic information on the driving directions from the final visiting node to the target position. However, if there is no learned information, or it the difference between the probabilities of arriving adjacent nodes is smaller than the threshold value (δ) set by the system, the node with the shortest path among the nodes adjacent to the target position is

selected as the final visiting node. The shortest path between all nodes can be found by using the information predefined in the TrafficApp.

Let the node corresponding to the current position of the vehicle be S, and let the GPS coordinates of the target position selected by the driver be TGPS. First, find the node N_T that is mapped to the TGPS by searching the hash search tree. Find the shortest distance from S to $N_{T,k}$ for all nodes $N_{T,k}$ ($0 \le k \le NT$ |) adjacent to N_T, including N_T. Here, $N_{T,0} = N_T$. Let Dist (a, b) be the shortest distance between a and b. The final visiting node N_{TPrev} for reaching the TGPS is determined by the following algorithm.

```
//RNetwork: Road Node Network
//A: A matrix of transition probability between every pair of Nodes
//SD: A matrix of shortest distance between every pair of Nodes
FindFVN(S, TGPS, N_T, RNetwork, A, SD) {
    find SP = N_{T,k} such that min_{k=0,...,|N_T|}{Dist(S, N_{T,k}) + Dist(N_{T,k}, TGPS)};
    find MaxProb = max_{k=0,...,|N_T|}{A_{SN_{T,k}}};
    MP = N_{T,k} of MaxProb;
    find MinProb = min_{k=0,...,|N_T|}{A_{SN_{T,k}}};
    if |MaxProb − MinProb| < δ then N_TPrev = SP;
    else N_TPrev = MP;
}
```

Finally, the TrafficApp generates a request message in the following format and requests traffic information for the target position.

TrafficRQST = {"RQST" || Target GPS || NID of Target Position || NID of Final Visiting Node || Timestamp};

= {"REQUEST" || $TGPS$ || N_T || N_{TPrev} || Timestamp};

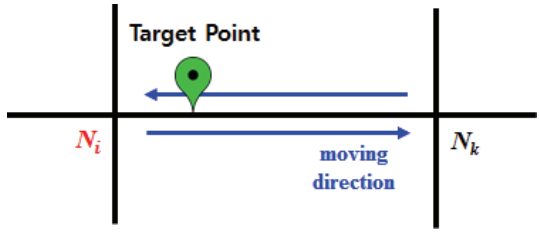


Fig. 5. Approach to target position.

In Fig. 5 above, the node being mapped is N_i because the target position is closer to N_i. The final node visited before reaching the target position is either N_i or N_k. In the case where the final visiting node is determined as N_k by the above algorithm, N_T will be equal to N_i and N_{TPrev} will be equal to N_k in the traffic information request message. Therefore, we determine the driving direction from N_k to N_i, and the TDMS searches for traffic information on the GPS closest to the target point among the traffic messages from vehicles moving from N_k to N_i and provides it to the vehicle. Second, in the case where the final visiting node is N_i, N_T will be equal to N_i and N_{TPrev} will be equal to N_i in the traffic information request message. This is a type of movement from N_i to N_i, and the traffic information on the GPS closest to the target position is searched and provided among the traffic messages from vehicles passing

by N_i while they are still near N_i. After all, the traffic information on the GPS closest to the target position is provided only from vehicles that pass by N_i, and which move from N_i to N_k. Therefore, the direction to reach the target position can be normally determined even if the N_T and N_{TPrev} values that correspond to the target position are the same.

5. Traffic Data Management System

In this section, we first describe a traffic information collection model that efficiently stores and manages traffic information collected from each vehicle.

5.1 TDMS System Configuration

Traffic data provided by each vehicle is divided into the TrafficMSG, which is a text message, and TrafficIMG, which includes acquired blackbox images. Because TrafficMSG and TrafficIMG have different transmission periods and file sizes, they are stored in separate distributed servers. All messages transmitted in real time from the TrafficAPP of vehicles are temporarily stored in a relational database (RDB) via a web server, and they are then transmitted to a data-processing pipeline (hereinafter referred to as DPP) to perform data refinement and classification. After correcting the traffic message, the TrafficMSG is stored in a message server (hereinafter referred to as MsgSV) and the TrafficIMG is stored in an image server (hereinafter referred to as ImgSV). The TrafficIMG is separated from the TrafficMSG and stored on a separate server as the TrafficIMG contains image files captured in the black box.

When the driver requests traffic information for a specific location through the TrafficAPP, the web server sends a search query directly to MsgSV and ImgSV for the requested traffic information message, and the search result is sent to the trafficAPP of the vehicle via the web server. The system configuration of the proposed TDMS is shown in Fig. 6.

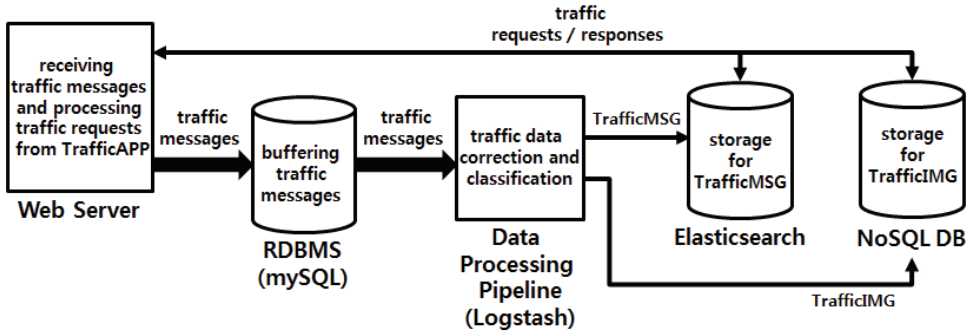


Fig. 6. Configuration of TDMS.

In this study, we employed MySQL for traffic message buffering and DPP for data correction and classification was implemented using Logstash. The message server for storing the TrafficMSG was implemented using Elasticsearch, while the server for storing the TrafficIMG was implemented using Mongo DB. We implemented the TrafficIMG storage server ted using a NoSQL-type Mongo DB to efficiently process a large number of files as the TrafficIMG includes image files.

5.2 Traffic Data Correction of DPP

Regardless of the type, all messages transmitted in real time contain GPS coordinates, CNID, which is the road node ID mapped to GPS coordinates, and PNID, which is the previously visited road node ID. CNID and PNID are the main factors that determine the driving directions of the vehicle. However, it cannot be guaranteed that CNID and PNID are necessarily adjacent to each other owing to the message transmission period. That is, the driving directions cannot be clearly determined in the case where PNID is not adjacent to the CNID. Therefore, DPP checks the adjacency between CNID and PNID for all collected messages. In the case where they are not adjacent to each other, then it finds the node closest to CNID based on PNID, corrects the PNID value, and transmits the corrected messages to each server. The correction strategy in this case is that the PNID value is updated to the node adjacent to CNID among the nodes on the shortest path between PNID and CNID. Because the shortest path between the nodes is already predefined and stored in the DPP, finding and updating nodes adjacent to CNID can be quickly handled. Messages with corrected PNID values are delivered to their servers according to their message type.

5.3 Traffic Data Management

Traffic messages are corrected through the DPP. They are stored and managed on the MsgSV or the ImgSV depending on the message type. The MsgSV stores values of <GPS, CID, PID, Speed, Timestamp, Type and MSG>. They are indexed by time and CID in sequence and stored in the server. The ImgSV stores the transmitted image files with values of <GPS, CID, PID, Speed, Timestamp>, and is indexed by date.

The traffic information message requested by the driver, TrafficRQST = {"REQUEST" || $TGPS$ || N_T || N_{TPrev}|| $Timestamp_R$}, is passed directly to the MsgSV and the ImgSV via the web server. Messages satisfying the following condition among the messages with the same date stored in each server are searched, and the latest traffic information is provided to the driver.

$$[(N_T == \text{CID}) \,\&\&\, (N_{TPrev} == \text{PID}) \,\&\&\, ((\text{Timestamp}_R - \text{Timestamp}) \leq TH_1) \,\&\&\, (\text{Dist}(TGPS, GPS) \leq TH_2)]$$

TH_1 and TH_2 are thresholds that are defined by the system. TH_1 refers to the threshold for the time difference, and TH_2 refers to the GPS distance difference. If there is no traffic data satisfying the above condition, it returns to the driver that there is no retrieved information.

5.4 Advanced Exploit of Proposed Model

In this study, every time a driver requests traffic information for a specific location, traffic information for the location is provided by predicting the driving directions. This can easily be improved by using more advanced automated driver-personalized traffic information services. That is, even if the driver does not request it manually, the TrafficAPP automatically predicts the driving directions and requests the TDMS for traffic information on the driving route ahead so that the driver can be provided with traffic conditions such as traffic congestion and accidents in advance for areas through which the vehicle is most likely to pass. This provides the driver with personalized traffic

information in advance according to the driving trajectory of each vehicle so that the driver can respond more quickly to traffic situations and make the required decisions for safer and more efficient driving.

6. Simulated Performance

In this section, we analyze the performance of the proposed model by performing various experiments. By focusing on the new main mechanisms proposed in this paper, we analyzed the performance of the hash search tree generation time and search time, real-time traffic information processing time of the TDMS, traffic information request message processing time, and the accuracy of driving direction prediction. In this experiment, we implemented the TDMS using one server, and we analyzed it as a performance criteria of one server. The server implementation utilizing the distributed server systems is beyond the scope of this paper, and will be discussed in future research. For the road information, in this study, we used the actual road node information of Seoul provided by the Ministry of Land, Infrastructure, and Transport, as well as the virtual nodes that were added. We used actual driving data obtained by two drivers driving for one month. The main experimental conditions and simulation parameters for the performance analysis are shown in Table 3.

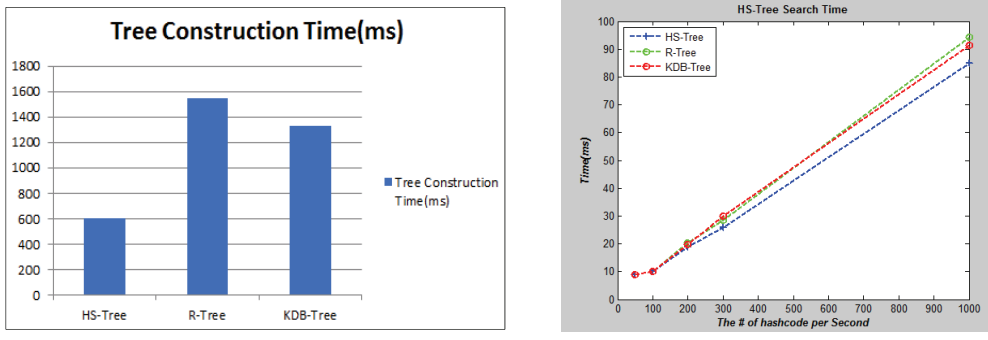
Table 3. Simulation parameters

Item	Parameter	Value
Road	The # of real nodes	7539
	The # of virtual nodes	38571
	The total # of nodes	46110
	Distance threshold for adding virtual nodes	100 m
	The minimum distance between two adjacent nodes	7.42 m
HS-tree	The minimum threshold / maximum threshold of a terminal	500 nodes / 800 nodes
	The length of hashcode	10 bits
TrafficAPP	Mobile device environment	Samsung Galaxy A8
	The # of cars	2
	The total # of trajectories per car	50
	The # of training trajectories per car	29
TDMS	System environment	Window OS, Intel i7, 64 bits, 3.4 GHz, 8G RAM
	The # of data records for sending to DPP	8000 data/min
	The simulated # of TrafficMSG per second	50, 100, 150, 200, 500, 1000
	The simulated # of TrafficIMG per second	50, 100, 150, 200, 500
	The simulated # of TrafficRQST per second	50, 100, 150, 200, 500, (1000)
Traffic message	TrafficMSG size	32 bytes
	TrafficIMG size	32 bytes + image size (300–900 kB)

In order to analyze the performance of the proposed hash search tree, we analyzed the search tree generation time and the tree searching time, and we compared the performance of the related research, the R-Tree and the kDB-tree. As shown in Fig. 7, the proposed HS-tree generation time is 600 ms, and the R-Tree generation time is 1550 ms. That is, the R-tree takes the longest time.

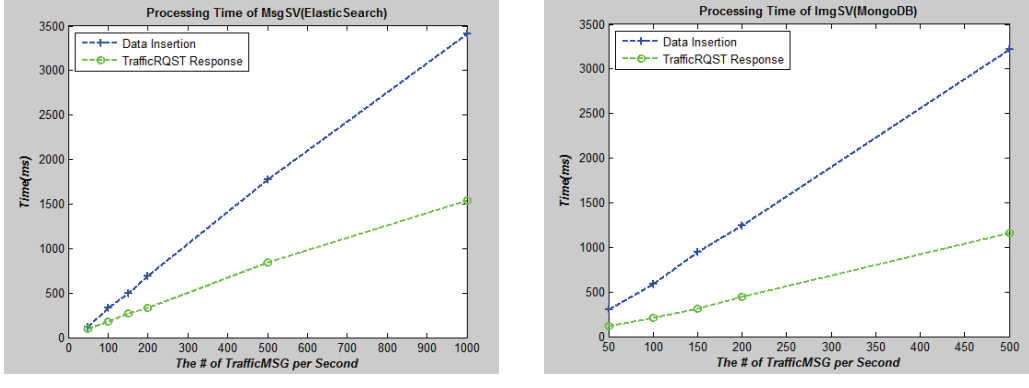
We analyzed the search time of the hash trees, was analyzed when the number of hash codes requested per second is 50, 100, 200, 300, and 1000, respectively. When searching for less than 100 hash codes per second, it took less than 10 ms for the three search tree modes. However, the search time of the proposed HS-tree is shortest, as shown in Fig. 7, when searching for more than 100 hash codes.

Next, we analyzed the time taken to store the TrafficMSG and TrafficIMG provided by each vehicle in the TDMS and to process the driver's request messages. The TrafficMSG is stored in the MsgSV implemented using ElastricSearch and the TrafficIMG is stored in the ImgSV implemented using MongoDB. For the experiment, when the randomly generated traffic messages in the smart device are transmitted to each server, the latency time to finish storing the messages is analyzed according to the number of messages per second. Because the TrafficIMG is transmitted less frequently than the TrafficMSG, the number of transmitted data messages per second is limited to 500. For the TrafficRQST processing time, the latency time to finish the response is analyzed according to the number of request messages per second. The driver sends a request message to ask for traffic information for a randomly selected area.



(a) (b)

Fig. 7. HS-tree construction time (a) and search time (b).



(a) (b)

Fig. 8. Latency time of TDMS: (a) MsgSV, (b) ImgSV.

As shown in Fig. 8 above, the user request information processing time is much shorter than traffic data insertion. For the TrafficMSG, 336 ms is the time required for 100 messages per s, and 688 ms for

200 messages. It takes less than 1 s to finish storing up to 300 messages. However it takes 1780 ms for 500 messages per s, and 3417 ms for 1000 messages, which means that a relatively long delay has occurred. For the response time for the driver request message, it takes 176 ms to send 100 messages, 329 ms for 200 messages, and 843 ms for 500 messages. It takes less than 1 s to send up to 600 messages. However, for 1000 messages, it takes 1540 ms, which means that the delay is long. The average size of the image file included in the TrafficIMG is 500 KB, which is much larger than that of the TrafficMSG. It takes a relatively long time to process the file and shows a somewhat longer latency time than the TrafficMSG. For the latency time to store the TrafficIMG in the ImgSV, 100 messages takes 591 ms, while 150 messages takes 950 ms. It takes less than 1 s for up to 150 messages. However, 1242 ms is the time taken for 200 messages and 3220 ms the time for 500 messages, which means that the delay is long. For the processing time of the TrafficRQST in the ImgSV, 214 ms is taken for 100 messages, 315 ms for 150 messages, 444 ms for 200 messages, and 1165 ms for 500 messages. It takes less than 1 s for up to 450 messages.

Finally, we analyzed the accuracy of the proposed driving prediction model. We analyzed the accuracy of three test sets: driving route with actual driving records (Test 1), driving route with no driving records (Test 2), and a driving route with a random mixture with/without driving records (Test 3). When the traffic information for the target point on the preset driving route is requested after predetermining the driving route, the accuracy was determined by providing the same driving route data as the preset driving route. The accuracy is shown in Table 4.

Table 4. Accuracy of driving trajectory estimation

	Test 1 (Known trajectories)	**Test 2 (Unknown trajectories)**	**Test 3 (Mixed trajectories)**	**Average**
Accuracy (%)	100	95	95	96.66

The above experiment results show that the driving prediction using driving records is 100% accurate and the driving prediction using the shortest distance also provides a fairly high accuracy when there are no driving records. From the experiments, it is appropriate to determine the driving directions based on the shortest distance.

7. Conclusion

In this paper, we proposed a traffic information service model that can collect traffic information in real time from each vehicle by utilizing a smart device in the existing vehicle, and which provides information on a specific location that the user wanted to reach. To manage traffic information more efficiently, we proposed a new hash search tree that can effectively search for road nodes that correspond to GPS coordinates by indexing traffic information based on road nodes. In addition, we proposed a probability model that predicts the driving directions by learning the driver's driving records in order to provide the driver with the traffic information based on the predicted driving directions. We performed experiments, which show that the proposed model is able to efficiently process the traffic information collected from each vehicle, and the driving prediction model provides

an accuracy of more than 96.7%. In future, we will continue to carry out research on various driver-customized services based on the driving direction prediction for each driver, and we will continue to study the implementation of a distributed TDMS environment for large-scale real-time traffic information processing.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. NRF-2014R1A1A3053491).

References

[1] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4-16, 1986.

[2] P. Mohan, V. N. Padmanabhan, and R. Rmajee, "Nericell: using mobile smartphones for rich monitoring of road and traffic conditions," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, Raleigh, NC, 2008, pp. 357-358.

[3] H. Zhu, Y. Zhu, M. Li, and L. M. Ni, "SEER: metropolitan-scale traffic perception based on lossy sensory data," in *Proceedings of INFOCOM*, Rio de Janeiro, Brazil, 2009, pp. 217-225.

[4] Z. Li, Y. Zhu, H. Zhu, and M. Li, "Compressive sensing approach to urban traffic sensing," in *Proceedings of 2011 31st International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, MN, 2011, pp. 889-898.

[5] R. K, Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "GreenGPS: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, And Services (MobiSys)*, San Francisco, CA, 2010, pp. 151-164.

[6] K. Chen and H. Shen, "RoadAware: learning personalized road information on daily routes with smartphones," in *Proceedings of 2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Waikoloa, HI, 2016, pp. 1-9.

[7] Y. Bao and W. Chen, "A personalized route search method based on joint driving and vehicular behavior recognition," in *Proceedings of 2016 IEEE MTT-S International Wireless Symposium (IWS)*, Shanghai, China, 2016, pp. 1-6.

[8] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, Boston, MA, 1984, pp. 47-57.

[9] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, Atlantic City, NJ, 1990, pp. 322-331.

[10] J. T. Robinson, "The K-D-B-tree: a search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, Ann Arbor, MI, 1991, pp. 10-18.

[11] Q. Liu, X. Tan, F. Huang, C. Peng, Y. Yao, and M. Gao, "GB-Tree: an efficient LBS location data indexing method," in *Proceedings of 3rd International Conference on Agro-geoinformatics*, Beijing, China, 2014, pp. 1-5.

[12] J. Yuan, Y. Zheng, W. Xie, and G. Sun, "T-Drive: enhancing driving directions with taxi driver's intelligence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 220-232, 2013.

Homin Han

He is an undergraduate student in Department of Software at Konkuk University. He is mainly interested in machine learning, image processing and distributed computing.

Soyoung Park

She received her B.S., M.S., and Ph.D. degrees in Department of Computer Science from Ewha Womans University, South Korea in 1998, 2000 and 2006, respectively. She has worked as a visiting scholar in University of Central Florida from 2006 to 2010. Currently, she is an assistant professor at the Department of Software, Konkuk University, Korea. Her research interests include cryptography, network security and privacy preserving ubiquitous computing.