CrowdNavi: Demystifying Last Mile Navigation With Crowdsourced Driving Information

Xiaoyi Fan, *Student Member, IEEE*, Jiangchuan Liu, *Senior Member, IEEE*, Zhi Wang, *Member, IEEE*, Yong Jiang, *Member, IEEE*, and Xue (Steve) Liu, *Member, IEEE*

***Abstract*—With detailed digital map of the transport network and even real-time traffic, today's navigation services provide good quality routes in the major route level. Once entering the last mile near the destination, they unfortunately can be ineffective and, instead, local drivers often have a better understanding of the routes there. With the deep penetration of 3G/4G mobile networks, drivers today are well connected anytime and anywhere; they can readily access information from the Internet and share information to the driver's community. This motivates our design of CrowdNavi, a complementary service to existing navigation systems, seeking to combat the last mile puzzle. CrowdNavi collects the crowdsourced driving information from users to identify their local driving patterns, and recommend the best local routes for users to reach their destinations. In this paper, we present the architectural design of CrowdNavi and identifies the unique challenges therein, particularly on identifying the last segment in a route from the crowdsourced driving information and navigate drivers through the last segment. We offer a complete set of algorithms to identify the last segment from the drivers' trajectories, scoring the landmark, and locating best routes with user preferences. We then present effective navigation algorithm to locate the best route along the landmarks for the last segment. We further realize the potential risks of attacks in crowdsourced systems and develop a multisensor cross-validation method against them. We have implemented the CrowdNavi app on Android mobile OS, and have examined its performance under various circumstances. The experimental results demonstrate its superiority in navigating drivers in the last segment toward the destination.**

***Index Terms*—Crowdsourced, internet of things, last mile navigation, mobile computing.**

I. Introduction

TODAY, online digital map services such as Google Maps are accessed by billions of users on a daily basis [1].

Manuscript received March 4, 2016; revised August 10, 2016; accepted September 14, 2016. Date of publication October 10, 2016; date of current version April 18, 2017. This work was supported by an NSERC Discovery Grant, a Strategic Project Grant, and E.W.R. Steacie Memorial Fellowship. The work of Z. Wang's is supported in part by the NSFC under Grant 61402247. Paper no. TII-16-0256. *(Corresponding author: X. Fan.)*

X. Fan and J. Liu are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: xiaoyif@sfu.ca; jcliu@sfu.ca).

Z. Wang and Y. Jiang are with the Department of Computer Science, Tsinghua University, Beijing 100084, China (e-mail: wangzhi@sz.tsinghua.edu.cn; jiangy@sz.tsinghua.edu.cn).

X. Liu is with the School of Computer Science, McGill University, Montreal, QC H3A 0E9, Canada (e-mail: xueliu@cs.mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2016.2616107

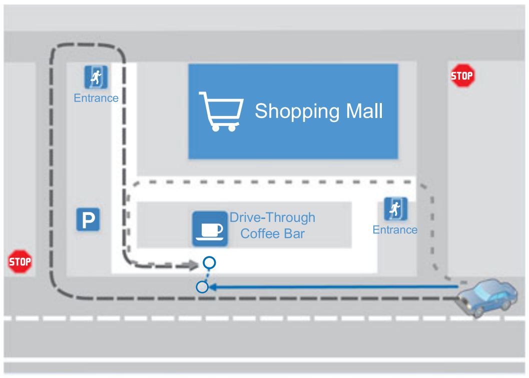


Fig. 1. Sketch of last segment from real-world instance.

Together with the advances of outdoor positioning services (GPS in particular), the navigation for drivers or pedestrians has increasingly been an essential application on smartphones or car consoles, which seeks to recommend the shortest or fastest routes using the digital maps and GPS. Real-time driving information, such as live traffic or construction locations has been incorporated as well. On the macroscale of a transportation network, the quality of the recommended routes is generally acceptable with state-of-the-art navigation services. Yet it is known that [2] the routes from the map-based services often fail to be agreed by local drivers, who have detailed knowledge and experience with local driving conditions.

In fact, quite often a driver is puzzled by the last mile near the destination, e.g., a building on a campus, where Google Maps or other similar navigation services provide little-detailed guidelines or simply fail. Fig. 1 shows a common example, where the destination is a drive-through coffee bar near a highway. Google Maps provides a straightforward route, as indicated by the solid line in Fig. 1. Although the destination appears in the line-sight for a driver in the last mile, s/he is effectively stranded to access the coffee bar given the road divider. Local drivers, however, will choose the two dashed lines, which are not straightforward but are accessible. In real-world driving scenarios, drivers will be busy in differentiating the landmarks, unclear shortcut roads, and available parking lots in a strange environment. These make searching the final destination even more difficult.

With detailed map of the transport network and even real-time traffic, existing navigation services provide fastest routes in the major route level; once entering the last mile, they unfortunately can be ineffective and, instead, local drivers often have a better

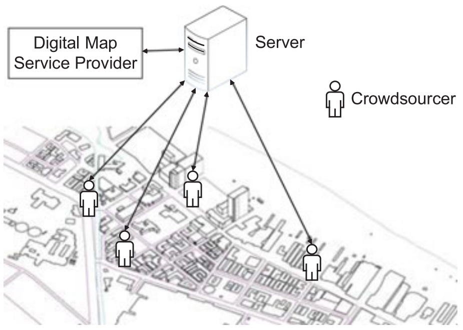


Fig. 2. Architecture of CrowdNavi.

understanding of the routes there. With the deep penetration of 3G/4G mobile networks, drivers today are well connected anytime and anywhere; they can readily access information from the Internet and share information to the community. These motivate our design of CrowdNavi, a complementary service to existing navigation systems, seeking to combat the last mile puzzle. CrowdNavi collects the crowdsourced driving information from users to identify their local driving patterns and recommend the best local routes for users to reach their destinations.

In this paper, we present the architectural design of CrowdNavi and identies the unique challenges therein, particularly on identifying the last segment in a route from the crowdsourced driving information and navigate drivers through the last segment. We offer a complete set of algorithms to cluster the landmarks from the drivers' trajectories, identify the entrance landmarks toward the last segment and evaluate the landmark preferences. We then present an effective navigation algorithm to locate the best route along the landmarks for the last segment. We further propose a multisensor cross-validation method to detect the emulated and malicious data generated by the attackers. We have implemented the CrowdNavi app on Android mobile OS and have examined its performance under various circumstances. The experimental results demonstrate its superiority in navigating drivers in the last segment toward the destination.

The rest of this paper is organized as follows. In Section II, we present an overview of CrowdNavi as well as the key challenges in its design. In Section III, we discuss the design principles and algorithms in different modules. In Section V, we propose a multisensor cross-validation method for the location authentication. Section IV shows the details of implementation in CrowdNavi. In Section VI, we compare CrowdNavi with Google Maps based on our real-world case study. Section VII provides a literature review and Section VIII concludes the paper.

II. CrowdNavi: Architecture and Challenges

Fig. 2 shows the overall architecture and workflow of CrowdNavi. Its user app will be installed on drivers' mobile devices, e.g., a smartphone or 3G/4G-enabled car consoles. If enabled, the app will run in the background, monitoring the moving trajectory of a car using GPS, and periodically reporting the location information to a backend server. The server accordingly maintains a database on the trajectory information of the app

TABLE I
Summary of Notations

d_j	destination j
T_j	trajectory set to destination d_j
T_i^j	trajectory i to destination d_j in T_j
n_i	trajectory point n_i
v_i	landmark
V	sets of landmark v_i
x_i	entrance landmark
X	entrance landmark set
L	last segments in trajectory set T_j
e_{ij}	landmark edge between v_i and v_j.
E	sets of landmark edge e_{ij}
$G = (V, E)$	routable landmark graph
u_k	user k
s_k	user sense of u_k
p_i	landmark popularity of landmark v_i
q_i	user preference on landmark v_i

users. When a user needs to find the route to a destination,[1] the request will be forwarded to the server, which will first identify the last segment closing to the destination. The route before the last segment will be recommended by an external map service (e.g., Google Maps). The last segment will then be calculated by the server using the database of the driving pattern gathered from the crowd.

We emphasize that the partitioning of the route is necessary given that the local drivers have the best (and sometimes the very only) knowledge for the last segment, as we have shown in the example in Section I. The challenges however are 1) how can we identify the last segment? and 2) how can we identify the best route toward the destination in the last segment? Such decisions are to be inferred from the massive trajectory information from the crowd, and therefore, should be automated with minimum user interference. In the next section, we will formalize this last segment navigation problem, and present a solution framework with trajectory data modeling, landmark preference mining, and favorite route recommendation, which consists of the core modules of CrowdNavi.

Before we proceed with the detailed solutions for the individual modules of CrowdNavi, we first summarize the key notations in Table I.

As mentioned, the server collects the trajectories of moving users from the app. For destination d_j, all trajectories toward this destination reported a trajectory set $T_j = \{T_1^j, T_2^j, \ldots\}$, in which each trajectory T_i^j is a sequence of GPS points, ended with d_j. That is, $T_i^j = (n_1, n_2, \ldots)$, where n_k is a tuple of the longitude, latitude, and time stamp of the corresponding GPS sample. Since we focus on last segment navigation toward the destination, we hereby omit the source index for ease of exposition. Note that different trajectories may intersect or overlap with each other, and a user may have multiple trajectories to destination d_j in the crowdsourced reporting context, e.g., the user visits his/her office (as the destination) every day and reports the trajectory for each visit.

[1]Note that, to use the navigation service, the user does not have to enable the app all the time to report the moving trajectory, although in this case the user will not help with populating the database.

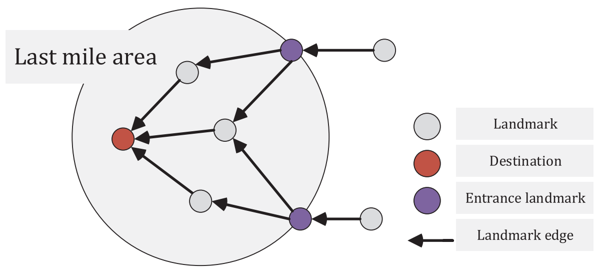


Fig. 3. Examples to illustrate the key terms.

To avoid dealing with the massive individual points on the trajectories, we rely on landmarks[2] for calculating routes as in early works [3]. Each landmark v_i is represented by a circle dot in Fig. 3, which stands for a geographic region that is of particular significance, e.g., the entrance to a campus or an intersection where many trajectories meet. We will discuss how a landmark v_i is identified with the crowdsourced driving information, as well as its popularity, user awareness, and preference later. Let $\Omega(v_i)$ denote the set of users whose historical trajectories appear on landmark v_i, and $\Theta(v_i)$ denote the set of the historical trajectories passing through landmark v_i. When forming landmarks, we will ensure each trajectory passes through at least one landmark. As such, we have $\bigcup_{vi\in V}\Theta(v_i)=T_j$. Between $\Omega(v_i)$ and $\Theta(v_i)$, for any user $u_k\in\Omega(v_i)$, we also define a mapping $f(u_k,v_i,t)=\{T_i^j|T_i^j\in\Theta(v_i,t),T_i^j.u=u_k\}$, where $T_i^j.u$ is the corresponding user of trajectory T_i^j. This mapping gives the set of trajectories passing through landmark v_i reported by user v_k.

If two landmarks v_i and v_k are directly connected by a certain trajectory, i.e., the trajectory sequentially pass through v_i and v_k without detouring, we say there is a landmark edge e_{ik} (arrow edges in Fig. 3) connecting them. We then have a routable landmark network $G=(V,E)$, where V is the set of all landmarks and E is the set of all edges. Given network G,a route from landmark v_i to destination d_j is formed by a sequence of connected landmarks, assuming that d_j is a landmark by default.

To formulate the last segment navigation problem, we first introduce the concept of entrance landmark set X (purple dot circles in Fig. 3), which is a subset of V, such that $\bigcup_{x_i\in X}\Theta(x_i)=T_j$ and its cost $\sum_{x_i\in X}C(x_i,d_j)$ is minimized. The cost $C(x_i,d_j)$ is the driving distance from x_i to d_j, i.e., the length of the corresponding trajectory segment. Intuitively, all the trajectories toward the destination will pass through at least one landmark in this set, and these entrance landmarks are the closest to the destination.

Definition 1 (Last Segment):The last segment L is a subgraph of $G=(V,E)$ that contains all possible routes from entrance landmark $x_i\in X$ to destination d_j, i.e., $L=\bigcup_{x_i\in X}\{r(x_i,d_j)\}$.

[2]We use the Google Maps Roads API to identify landmarks. The API takes up to 100 GPS points collected along a trajectory and returns a similar set of data with the points snapped to the most likely roads the vehicle was traveling along. Intuitively, each snapped point represents a critical location on the road, e.g., an intersection, and we set it as a landmark.

As mentioned, the trajectories from the crowd can be highly diverse, particularly around the destination. The last segment navigation therefore is to identify the best entrance landmark and the route toward the destination, as follows:

Problem Definition. Given a user query to destination d_j, identify the entrance landmark set X, and find the favorite route $r^*(x_i,d_j)$ for any $x_i\in X$ from the last segment L.

III. Identifying and Navigating the Last Segment

The navigation problem may be solved by recruiting experienced users to manually mark the last segments, which however can be labor intensive and time consuming, and is hardly scalable with massive user bases. The preference of the manual marking is also questionable. Instead, our CrowdNavi explores the rich information inside the trajectory data from its users, trying to identify the last segment as well as the best route toward the destination from the sheer amount of data.

A. Identifying the Last Segment

We start from the preprocess step to construct a routable landmark graph for identifying all landmarks from trajectory dataset. We discover geographical areas and divide a geographical range into disjoint clusters with a spatial clustering algorithm. DBSCAN [4] is a notably representative, which, given a set of points, groups together points that are closely packed together. Based on the routable graph $G=(V,E)$, we extract out all the historical trajectories T_j to destination d_j and search the last segment L with the spatial and social-community properties in the destination area. The recognition for an entrance landmark depends on two landmark parameters $\Theta(v_i)$ of landmark v_i and driving distance $C(v_i,d_j)$ from v_i to destination d_j. These landmarks can be regarded as *entrance landmark*, when they satisfy the following two conditions. One is that $\Theta(v_i)$ of landmark v_i is exceeding most other landmarks, which means this landmark v_i is located on essential routes to destination d_j. The other condition is that the sum of driving distance $C(v_i,d_j)$ in entrance landmark set should be as small as possible. In most cases, the entrance landmark occurs when people drive or walk away from some certain roads and search for a specific entrance, or are confused by the surrounding environment. Our objective is to find last segment L for destination d_j based on trajectory dataset T_j. With definition 1 for last segment, we formulate this problem of searching entrance landmark of last segment as a minimum weighted set cover problem. Each landmark $v_i\in V$ has a variable $\beta_i\in\{0,1\}$ which defines whether v_i is in the set of entrance landmark X

$$\begin{aligned}\text{Minimize}\quad & \sum_{v_i\in V} C(v_i,d_j)*\beta_i\\ \text{s.t.}\quad & \bigcup_{T_k\in\Theta(v_i)} T_k\geq T_j\quad \forall v_i\in V\\ & \beta_i\in\{0,1\}\quad \forall v_i\in V. \end{aligned}\tag{1}$$

The problem of finding a minimum weighted set cover is a classical optimization problem with approximation algorithm-s [5]. With the entrance landmark set $X=\{v_i|\beta_i=1,v_i\in V\}$,

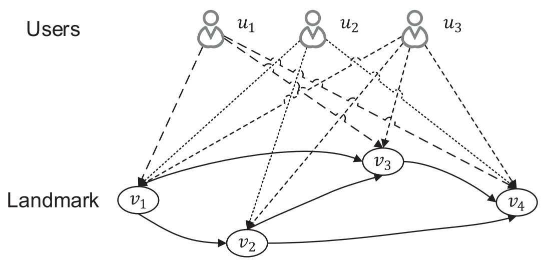


Fig. 4. Landmark scoring model.

we can obtain the last segment set $L = \bigcup_{x_i \in X} \{r(x_i, d_j)\}$ with all possible routes to the destination d_j based on Definition 1.

B. Scoring Landmark

We propose a landmark scoring model to quantify the landmark preference q_i, as Fig. 4 shows. Fig. 4 illustrates the main idea of landmark scoring model, which is inspired by [6]. There are two kinds of nodes, i.e., users and landmarks, where a user has passed through many landmarks and a landmark has been passed through by many users. For example, user u_1 passed through landmarks $v_1 \to v_3 \to v_4$, where u_1 points to landmark v_1, v_2, and v_3. Thus, a good user can point to many good landmarks, and a good landmark is pointed by many good users. We propose the *landmark preference* and the *user sense* for each landmark and user, respectively. Yet, the user sense and the landmark preference has a more complicated mutual reinforcement relationship than [6], which will be clarified later. Using a power iteration method, the user sense and the landmark preference can be calculated.

Definition 2 (Landmark Popularity): We define the popularity p_i for landmark v_i as the portion of the users whose trajectories appear on landmark v_i. That is,

$$p_i = \frac{|\Theta(v_i)|}{|T_i|}, v_i \in V \tag{2}$$

where $\Theta(v_i)$ is a set of trajectories passed through landmark v_i and T_j is the set of trajectories toward destination d_j. The popularity p_i is the fraction of users who like the landmark.

Existing works [7], [8] have used popularity as the only evaluation metric for landmarks, which may neglect superb ones due to massive noises or bias against unpopular ones. 1) There can be some local people who know well about every landmark around their home. Although they choose their favorite routes, many people who are unfamiliar with this area also provide many not-so-good historical trajectories as noises, flooding the good routes from the local people. Without the users' experience evaluation, it is difficult to dig out the good routes from the trajectory dataset. 2) Consider a new road that has just been built. We assume that the road is of very high preference and anyone who knows this road agrees that the road should be ranked highly. Even so, since the road is new, there exist only a few historical trajectories on this road. Thus, the landmarks on this road with very low priority will be never recommended. Since the system does not recommend it, few users ever know this landmark, and so the popularity of this landmark increases very slowly. It takes a very long time for new landmarks to be discovered by the users and the system. Based on above discussions, we propose a concept of *user preference*, where the challenge is that different people may have completely different preference judgement on the same landmark and the judgement on landmark also may change as time goes by.

Our main idea to calculate the landmark preference is based on the fact that a preferred landmark will increase its popularity much more rapidly than others, when a large fraction of highly experienced (i.e., good user sense) drivers redirect to the landmark. Therefore, by observing the increase of landmark popularity with user sense, we can now derive the landmark preference evolution function over time under our model. When we know the current landmark popularity p_i, we can estimate how many new users will pass through landmark v_i. Therefore, we define the preference q_i of a landmark v_i as follows:

Definition 3 (User Preference): User preference q_i gives the degree on how the users in the community like the landmark v_i or not. Specifically, user preference q_i on landmark v_i will increase, if many local users with high user sense have considerable trajectories on landmark v_i

$$q_i = \sum_{u_k \in \Omega(v_i)} \frac{\mathrm{d}f(v_k, v_i, t)}{\mathrm{d}t} \cdot s_k + p_i \tag{3}$$

where $f(u_k, v_i, t)$ is the frequency of user u_k that passed through landmark v_i at time *t*; $\Omega(v_i)$ is the set of users who passed through landmark v_i;[3] and s_k is user sense of user u_k who is aware of this landmark v_i.

Definition 4 (User Sense): The sense of user s_k is the sum of landmark preference q_i, which represents the knowledge of user u_k in last segment

$$s_k = \sum_{v_i \in u_k} q_i. \tag{4}$$

The range of s_k value [0, 1], where the initial value is 0. When the user knows each landmark, the value is 1.

C. Last Segment Routing

With the last segment L and the set of entrance landmarks *X*, we can build a vertex-weighted last segment graph, which is $G' = (V', E', w)$, $V' = \{v_i | v_i \in L\}$, $E' = \{e_{ij} | e_{ij} \in L\}$, and $w = \{q_i | v_i \in V'\}$. Note destination d_j serves as a landmark (in V') by default. We now proceed to find a favorite route $r^*(x_i, d_j), x_i \in X$ among all possible routes, so as to maximize the weight of the minimum-weight vertex in the route and with least hops. The challenge is: 1) How can we select the optimal landmark from the entrance landmark set *X*? 2) How can we find a favorite route $r^*(x_i, d_j)$ in a vertex weighted graph with antiparallel edges? We address them by a transform on vertex weighted graph, as shown in Fig. 5. 1) Multiple-sources single-sink problem can be converted to a single-source single-sink problem by introducing a dummy source vertex v_0 into V' that

[3]For ease of computation, the user preference qi has been normalized by the corresponding sum of total user preference.

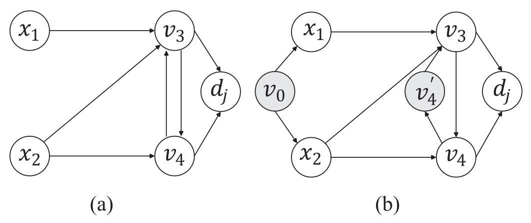


Fig. 5. Adding dummy source vertex and removing antiparallel edges.

is connected to the original source vertices $x_i \in X$. In Fig. 5(b), v_0 is added as a dummy source vertex and points to x_1 and x_2. The dummy source vertex v_0 gets a weight that is equal to the total weights of all the source vertices $x_i \in X$. 2) Since there are antiparallel edges in the vertex weighted graph, we must transform the network into an equivalent one without antiparallel edges. We arbitrarily select one from antiparallel edges and split it by inserting a new vertex. We also set the weight of new vertex to the weight of the original source vertex, i.e., v_4' has the equal weight with v_4. The resulting network satisfies the property that if one edge is in the network, the reverse edge is not.

This modification enables us to solve it as finding a shortest path of maximum bottleneck in weighted vertex graph [9]. Let $h(v_0, v_i)$ denote the hops (unweighted distance) from v_0 to v_i and let $sc(v_0, v_i)$ denote the maximum bottleneck weight of a route from v_0 to v_i. Let $r(v_i)$ denote route from v_0 to v_i, which is a sequence vertex. We start by setting $h(v_0, v_0) = 0$, $sc(v_0, v_0) = w_0$, $r(v_0) = \emptyset$ and place v_0 in the queue Q. All other vertices are initialized with $h(v_0, v_i) = \infty$ and $r(v_i) = \emptyset$. When v_i leaves the queue, for each $e_{ij} \in E$ the following is performed. If $h(v_0, v_j) = \infty$, we set $h(v_0, v_j) = h(v_0, v_i) + 1$, $r(v_j) = r(v_i) + v_j$, place v_j in the queue Q and set $sc(v_0, v_j) = \min\{sc(v_0, v_i), w_j\}$. Otherwise, there exists a route $r(v_j)$ that arrives v_j from v_0. If $\mathrm{sc}(v_0, v_j) < \min\{\mathrm{sc}(v_0, v_i), w_j\}$, we choose the new route $r(v_i) + v_j$. If $\mathrm{sc}(v_0, v_j) == \min\{\mathrm{sc}(v_0, v_i), w_j\}$, we tend to favor the route with fewer hops.

IV. Implementation

We dedicate this section to investigations of how CrowdNavi performs in real-world systems. Fig. 6 shows the workflow between the mobile client app and the CrowdNavi server. The system follows a simple client–server architecture. The processes run offline and periodically execute our algorithms to find the favorite routes based on the trajectory data. We deploy the database on the servers to store the trajectories and last segment information. CrowdNavi Backend module on the server allows mobile clients to query the route. The route requests and routes are delivered by JavaScript Object Notation, which is easy to be extended to other platforms. Fig. 7(a) shows the interfaces of CrowdNavi in a mobile browser running on Google Nexus 5 Android smartphone, which is implement by JavaScript and similar to Google Maps App. We have implemented CrowdNavi app in Android OS 5.1.1 working together with Google Maps

Algorithm 1: CrowdNavi routing algorithm.

```
Input: $G' = (V', E', w)$
Output: the favorite route $r^*(x_i, d_j)$
 1: $h(v_0, v_0) = 0$
 2: $sc(v_0, v_0) = w_0$
 3: $Q = \{v_0\}$
 4: for each $v_i$ in $V'$ do
 5:     $h(v_0, v_i) = \infty$
 6:     $r(v_i) = \emptyset$
 7: end for
 8: while $v_i$ leave $Q$ do
 9:     for each $e_{ij}$ in $E$ do
10:         if $h(v_0, v_j) = \infty$ then
11:             $h(v_0, v_j) = h(v_0, v_i) + 1$
12:             $r(v_j) = r(v_i) + v_j$
13:             insert $v_j$ into queue $Q$
14:             $sc(v_0, v_j) = \min\{sc(v_0, v_i), w_j\}$
15:         else
16:             if $sc(v_0, v_j) < \min\{sc(v_0, v_i), w_j\}$ then
17:                 $sc(v_0, v_j) = \min\{sc(v_0, v_i), w_j\}$
18:                 $h(v_0, v_j) = h(v_0, v_i) + 1$
19:                 $r(v_j) = r(v_i) + v_j$
20:             else if $sc(v_0, v_j) == \min\{sc(v_0, v_i), w_j\}$
                then
21:                 if $h(v_0, v_j) > h(v_0, v_i) + 1$ then
22:                     $h(v_0, v_j) = h(v_0, v_i) + 1$
23:                     $r(v_j) = r(v_i) + v_j$
24:                 end if
25:             end if
26:         end if
27:     end for
28: end while
29: Return $r(d_j)$
```

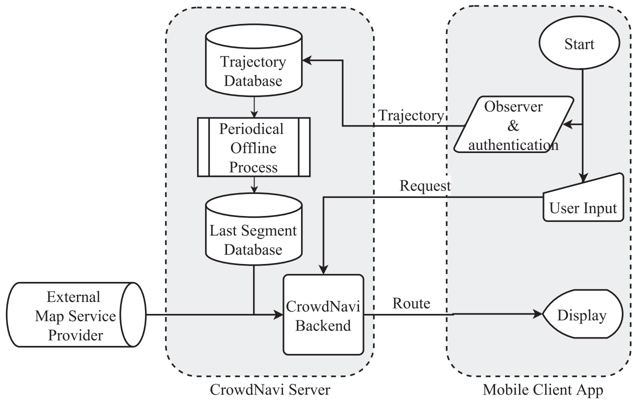


Fig. 6. CrowdNavi workflow.

Android API[4] as Fig. 7(d) shows, which is based on JAVA in the android-studio programing environment. CrowdNavi also runs in background as an observer to sample and buffer the user's real-time GPS streams.

[4]https://developers.google.com/maps/documentation/android/

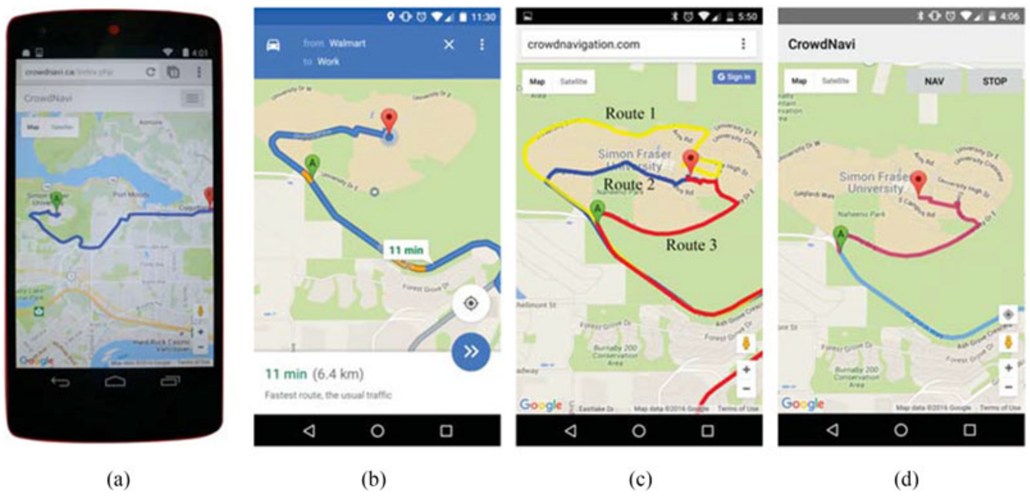


Fig. 7. Implementation and case studies of CrowdNavi. From Marker A to the destination, 5% users drive on Route 1, 20% users drive on Route 2 following Google Maps and 75% users choose Route 3. (a) LG Nexus 5, (b) Google Maps, (c) trajectory dataset, and (d) CrowdNavi.

GPS and high-speed Internet connectivity are power hungry [10], where the aggressive usage of GPS and Internet can drain the battery completely within a few hours. CrowdNavi tackles the challenges of energy consumption in two folds. 1) It adopts the design of SensTrack [10], which selectively executes a GPS sampling using the information from the acceleration and orientation sensors. 2) To minimize the traffic, it includes an optimized implementation of packets caching and data compressing on the mobile client [11]. When the high-speed wireless connection is available, CrowdNavi adopts a "race to sleep" strategy to upload the trajectories data in a batch to the server. If transmission failures occur, CrowdNavi will store data in the local storage and an available Wi-Fi connection can trigger the batch data transmission.

CrowdNavi servers are deployed on Amazon EC2 cloud platform, which are in charge of responding to users and data processing, as shown in Fig. 6. The trajectory data from users can be inserted into the trajectory database. Last segment and landmark preference calculation are the computation bottlenecks in our approaches, since it involves algorithm with power iterations. This algorithm further invokes the costly bottleneck path calculation for each last segment. Those iterative algorithms are executed periodically offline, which targets to get a relatively stable landmark preference. In practice, the landmark preference for each last segment can converge in tens of iterations. As we discussed, CrowdNavi complements Google Maps in last segment, which provides a route from entrance landmark to destination and relies on Google Directions API[5] for searching rest part of route.

V. Location Authentication in CrowdNavi

Crowdsourced systems are vulnerable to mischievous or malicious users who are seeking to disrupt the system [12]. For example, malicious users supported by the competitors can forge their physical locations to mislead the navigation system. Unfortunately, there are no widely deployed techniques to authenticate the location of mobile devices. CrowdNavi, as a crowdsourced service, also faces a number of security vulnerabilities and has to make efforts toward location authentication. The significant attacks against crowdsourced maps application can be achieved by using widely available mobile device emulators that run on computers. Most mobile emulators run a full OS (e.g., Android, iOS) and simulate hardware features, such as camera and GPS. Attackers can generate user actions on CrowdNavi, such as clicking buttons and typing text, and feed predesigned GPS sequences to simulate location positioning and device movement. By controlling the timing of the GPS updates, they can simulate any movement speed of the emulated devices. By exploiting this vulnerability, attackers can create fake traffic hotspots and misleading routes at any location on the map.

The main idea is that the messages describing the device moving from inertial sensors should keep consistent with the data from the GPS device, and CrowdNavi uses the cross-validation method among the multiple sensors to verify the physical status of mobile devices. The inertial sensors for localization have been widely explored [13]. CrowdNavi holds the assumption that the multisensor cross-validation method requires the messages from inertial sensors in the millisecond level and it is difficult for attackers to generate such accurate and dense data. GPS locations in the short interval (5–10 s) can be precisely predicted and matched with the estimated trajectory, where a sequence of inertial sensor readings are used to generate the estimated trajectory. The smartphone with Inertial Navigation Systems contains accelerometers, gyroscopes, and magnetometers, which can track orientation and position changes and the users' absolute heading. The messages from those sensors are extremely dense and difficult to be emulated, which increase difficulties and limit the ability to amplify the potential damage incurred by any single attacker. Since the precise location inference by the inertial sensors is possible in 1 or 2 min [14], we use the messages from inertial sensors for the location

[5]https://developers.google.com/maps/documentation/directions/

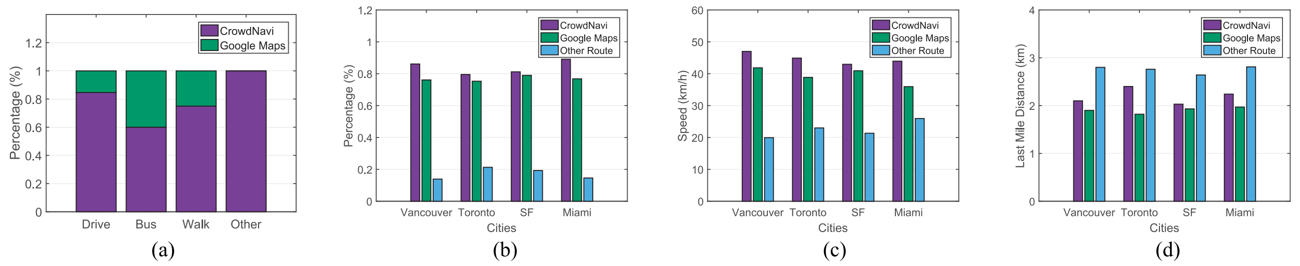


Fig. 8. Experimental results to compare CrowdNavi and Google Maps. (a) Web survey comparison of CrowdNavi against Google Maps, (b) percentage of the candidate route chosen by the drivers, (c) average speed of the candidate route, and (d) average distance of the candidate route.

authentication in CrowdNavi, where the estimation errors have not yet aggregated in such a short interval (5 s).

CrowdNavi depends on the known GPS position and velocity at *t*–1 time as the initial status and then uses a set of messages from inertial sensors during [*t*–1, *t*] time to compute the trajectory in a few seconds. Assuming the sensor readings are correct and sensitive, CrowdNavi applies a physics approach to estimate the coming device location intro and compare it with the GPS point at *t* time. In particular, CrowdNavi gets the device's moving orientation by Android API(SensorManager.getOrientation()), as well as the acceleration along this moving direction. When the estimated trajectory frequently mismatches with the roads on the map or the inertial inferred location is often inconsistent with the coming GPS point, CrowdNavi grades this device as an attacker and its trajectory as discarded.

VI. Evaluation

This section presents the evaluation of our CrowdNavi system. CrowdNavi provides the best route for each destination in last mile area, which is evaluated against the result provided by the Google Directions API.[6]

A. Case Studies

Before conducting the quantitative performance evaluation, we provide one instance as case studies to demonstrate the effectiveness of CrowdNavi system. The destination of the case study is one building of our workplace on the campus. We first search the Google Maps, which provides one route as shown in Fig. 7(b). We then set up an experiment and collect the routes from 20 volunteers, who are familiar with the roads and drive with their own preferences. As shown in Fig. 7(c), their routes match well with those recommended by Google Maps in the city's major road level. However, when we are near the starting point or close to the destination, the routes chosen by the volunteers however become quite diverse and deviate significantly from the Google recommended routes. The route in Fig. 7(b) is not a good choice with various weaknesses, including narrow roads with many crossroads, many people walk on the campus road, coinciding with the bus lines, and many reserved street parking for campus service vehicles. These volunteers are much more familiar with the exact road conditions on the campus, including the intersections, backstreets, and roadside parking slots; the routes in CrowdNavi are therefore often better than the recommendation from Google Maps. As shown in Fig. 7(d), CrowdNavi recommends a highway on the edge of the campus, where our field check suggests that this route is highly practical and reasonable.

We have conducted a user questionnaire survey to further explore the results. As Fig. 8(a) shows, most of the existing studies on navigation services collect the feedback from volunteers to derive the user experience. Trying to directly understand the QoS of CrowdNavi and real preferences of users on different routes, we have invited local people to fill in our web survey. Ninety people have participated in the survey, where 96.67% work or study in our campus and 73.33% know the destination in Fig. 7. The survey contains a series of single-choice questions plus several questions on insensitive personal information. The result is gratified that only 30% of users will not choose our route in Fig. 7(d). Of the users who do not select our route in Fig. 7(d), 78% participants take buses daily to campus and are not aware of the route in Fig. 7(d).

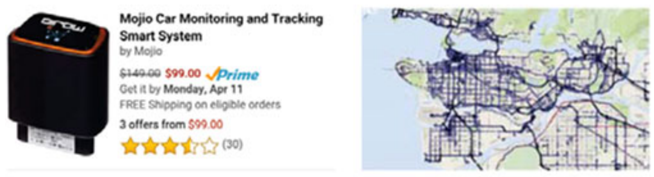
B. Large-Scale Experiments

It is difficult to directly evaluate whether a customized route provided by CrowdNavi for a real user is the best one due to the following two reasons. First, a user can only drive on one route at a given time. The user would not know whether other routes are better than the driven one. It is not reasonable to have a different user to travel on another route simultaneously with their different drive behaviors and knowledge to the evaluation. Second, it is not reasonable to request a single user to drive two different routes separately, since the user can learn from their past driving experiences. The route driven later will benefit from the first test.

We have closely collaborated with Mojio,[7] a leading company in vehicle status monitoring, to analyze their user logs, which have been used as crowdsourced data in the experiments. Mojio connects the vehicles into Internet with a customized device as shown in Fig. 9(a), which is installed on the vehicle on-board diagnostics (OBD-II) port. When a user starts the engine, the vehicle data will be sent to the Mojio's cloud platform. The

[6]https://developers.google.com/maps/documentation/directions/

[7]https://www.moj.io/


Mojio Car Monitoring and Tracking Smart System
by Mojio
$149.00 $99.00 ✓Prime
Get it by Monday, Apr 11
FREE Shipping on eligible orders
3 offers from $99.00
(30)

(a) (b)

Fig. 9. Sketch dataset and device from Mojio. (a) device from Mojio and (b) trajectories in Vancouver.

TABLE II
MOJIO DATA

	Records	trajectories	Vehicle	Destinations
Vancouver	459 922	4228	191	118
Toronto	194 789	1626	133	38
San Francisco	148 116	1329	106	45
Miami	720 247	5806	256	187

trace data record such real-time information as the local time, vehicle model, GPS position, etc. The data that we have used include 922 vehicles with 6 154 134 records, spanning from 13 August 2015 to 27 August 2015. In the dataset, a GPS trajectory is represented by a sequence of time-stamped points, which contain the information of latitude, longitude, speed, and so on. Fig. 9(b) plots the overview of Mojio data in Vancouver, where each dot is one GPS record. More details are shown in Table II. We have located 29.87% destinations in the Mojio's data with the last mile puzzles, where the criterion is that each destination has at least ten different trajectories from the entrance landmark. The number (almost 30%) suggests that the last mile with puzzles is indeed quite common in the real-world driving. We evaluate the performance of CrowdNavi based on the Mojio's data.

To examine the performance of CrowdNavi, we have conducted experiments in four cities in North American, involving Vancouver, Toronto, San Francisco, and Miami. As Table II shows, we collected 686 vehicles logs with 12 989 trajectories. Seventy-five percent trajectories can be regarded as the training dataset and the remaining data are used as the validation dataset. Fig. 8 shows that CrowdNavi perform well in the large-scale experiments. In particular, Fig. 8(a) illustrates the percentage of the candidate route chosen by the drivers. The results demonstrate that the route of CrowdNavi is more preferred than Google Maps and other routes. We also observe that there are many overlaps between the routes selected by CrowdNavi and Google Maps, which means that CrowdNavi and Google Maps recommend the same route in many cases. We further explore the difference between CrowdNavi and Google Maps in Fig. 8(b) and 8(c). Fig. 8(b) plots the average speed the users traverse in last mile area, and Fig. 8(c) shows the average distance the users traverse in last mile area. The routes provided by CrowdNavi have higher speed than Google Maps, while the routes of Google Maps have less distance. The above figures together demonstrate the preference of most drivers in last mile that drivers would prefer the routes with higher speed rather than the shortest path. The higher speed routes usually contain fewer STOP signs, fewer turns, and less light traffic, although those routes may have longer than the shortest one. Although the difference of the spent time is trivial in the last mile driving, CrowdNavi usually provides an easier accessible route than Google Maps, e.g., CrowdNavi can recommend a correct road side to avoid crossing the streets. Moreover, we analyze the unpopular routes in the logs, e.g., local people occasionally may drive in the detours, or the inexperienced drivers lost directions in the last mile.

C. Simulations

We conduct simulations to examine the effectiveness of the proposed CrowdNavi routing algorithm. We compare CrowdNavi with the HITS-based [6] and popularity-based [15] approaches. In the simulation, there are a total of 50 local drivers navigating toward one destination, along which there has already been an old road, and a newly built one is also available. We set the newly built road to be much better than the old one, and as such all drivers will prefer the new road. We then generate user trajectories with the interval of one day. The drivers randomly drive through the road and arrive at the destination up to five times each day. Once a driver is aware of the new road, he/she will choose it accordingly. Fig. 10(a) shows that the rating of the new road increases rapidly with CrowdNavi in two days. The popularity-based approach however only slowly increases the rating, being ineffective to popular route changes. The HITS-based approach cannot utilize the temporal features of the historical trajectories, either. In particular, the ratings of two routes are equal to 0.5, since the experienced drivers have many historical trajectories on both two roads and such an approach can hardly differentiate the users' preference changes.

In Fig. 10(b), we evaluate the effects of noises, i.e., the incorrect or malicious users' trajectories. We generate the incorrect trajectories based on a Pareto distribution [16] with parameter $\alpha = 1$, yet our approach can easily adapt to any popularity distribution once it is known or can be estimated. Through the simulated experiments, Fig. 10(b) demonstrates that CrowdNavi is quite immune from noises and provides a proper route under the effects of noises. We also observe that the HITS-based approach can hardly eliminate the effects of interference and mistakenly rate the road with a high preference.

D. Accuracy of Attack Detection

We examine the multisensor cross-validation method by providing the forged and real-world data to CrowdNavi apps, where each category contains 1000 trajectories. The forged data as malicious attacks is generated by randomly selecting the inertial messages from the real-world trajectories and feeding those massive data with the predesigned GPS sequences to CrowdNavi. Fig. 10 shows the percentage of trajectories marked as distrusted in two datasets, which demonstrates the attacks detection mechanism in CrowdNavi is highly effective. The percentage of detected malicious devices in the forged dataset is close to 1, indicating CrowdNavi can identify almost all attacks even after the attacker provide the information of the

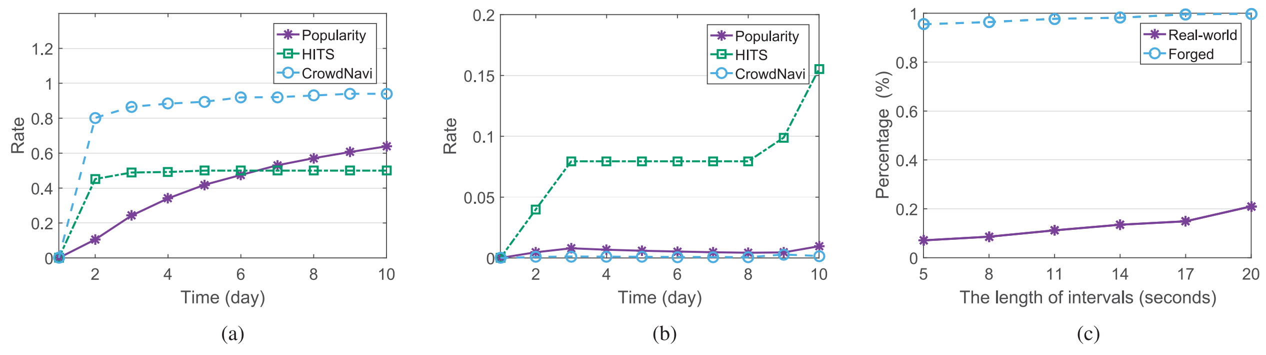


Fig. 10. Experimental results of CrowdNavi. (a) Newly built road, (b) noise road, and (c) percentage of trajectories as discarded.

inertial sensors. Interestingly, the length of intervals is an important factor for the accuracy of attack detection. Due to the fact that the estimation error growth with time and traveled distance, CrowdNavi has to frequently inspect the messages of the device among the multiple sensors. Otherwise, the real-world positive devices would be mistakenly graded as distrusted.

VII. Related Work

In this section, we review the related work and clarify the differences as compared to CrowdNavi.

Route planning plays an important role in our paper. Finding the shortest path has been extensively studied for over fifty years. For route planning, time-dependent shortest path problem [17] regards the travel time of an edge as a single-valued function. Chen *et al*. [18] focus on finding the top-k best connected trajectories that are geographically optimal to connect user-given locations. T-drive [3] and the followup work [19] finds the fastest routes to a destination through gradually learning users' driving behavior from historical GPS logs reported by a large collection of Taxis. In [7], the problem of popular route planning without road network information is investigated and it has been demonstrated that the most popular route between two given locations can be inferred using the turning probability of each intersection. In [20], the most frequent path problem with user-specified time periods has been examined, which can recommend time-dependent routes in arbitrary time periods specified by the users. Wei *et al*. [8] search for popular and attractive travel routes, aiming at finding the top-k routes from uncertain trajectories. CrowdNavi focuses on the last segment navigation with crowdsourced driving information, serving as a complement to existing online map and routing planning systems. Our evaluation also suggests that, in this context, the driving data crowdsourced from local drivers could be more effective than those collected from Taxis.

The idea of crowdsourcing has been popular in the map building, navigation and localization, particularly with the advancement of modern smart mobile devices and the deep penetration of 3G/4G mobile networks [21], [22]. Google and TomTom have leveraged crowdsourcing for online map update and maintenance, where user-submitted changes can be integrated into their map products after manual review.

For example, Google Local Guide[8] encourages users to help others with their local experience, e.g., writing reviews or rating restaurants. Waze,[9] another popular community-driven application from Google, accepts diverse traffic information and incident reports from general users to issue accident alerts and identify preferred routes with consistently updated live maps. Hi-Park[10] as a startup is developing a crowdsourced last mile navigation application, which enables users to receive step by step guidance to their desired destination. OpenStreetMap [23] has attracted millions of contributors and users to establish and maintain the on-line map infrastructure by its volunteer community.

Mobile location authentication and defending against forged GPS are challenging. The trusted software/hardware modules on mobile devices [24] are used to prevent GPS spoofing by enforcing strong location authentication; however, this is ineffective when attackers have physical access to the phone. Other solutions authenticate user locations using wireless in-frastructures, e.g., Wi-Fi APs [25], thus devices must come into physical proximity to these infrastructures to be authenticated. Recently, researchers have proposed peer-based methods to authenticate colocated mobile devices [12]. Different from existing studies, CrowdNavi utilizes a multisensor cross-validation method to increase difficulties for attacks, instead of directly authenticating a devices physical location.

Our design of CrowdNavi is motivated by these crowdsourced map services and related trip recommendation services. Yet we focus on the last segment navigation, not the overall trip, and closely examine its unique challenges. We also seek to fully automate data collection and analysis with minimum human interference.

VIII. Conclusion

In this paper, we have presented the architectural design of CrowdNavi as a supplementary to the current digital map services. The unique challenges therein have been illustrated, particularly on identifying the last segment in a route from the crowdsourced driving information and guiding drivers through

[8]https://www.google.com/local/guides/
[9]https://www.waze.com/
[10]http://www.hi-park.co/

the last segment. We have offered a complete set of algorithms to cluster the landmarks from the drivers' trajectories, identify the entrance landmarks toward the last segment, and evaluate the landmarks preferences. We then presented the navigation algorithms to locate the best route along the landmarks toward the destination. Moreover, we proposed and validated a suite of techniques to effectively detect the malicious attacks and prevent the negative consequence in CrowdNavi.

References

[1] "Google i/o 2013 session (google maps: Into the future),". [Online]. Available: https://www.youtube.com/watch?v=sBAd89C4Q8Q/

[2] V. Ceikute and C. S. Jensen, "Routing service quality–local driver behavior versus routing services," in *Proc. 14th IEEE Int. Conf. Mobile Data Manage.*, 2013, pp. 97–106.

[3] J. Yuan *et al.*, "T-drive: Driving directions based on taxi trajectories," in *Proc. 18th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2010, pp. 99–108.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *ACM SIGKDD Knowl. Discovery Data Min.*, vol. 96, no. 34, pp. 226–231, 1996.

[5] M. Karpinski and A. Zelikovsky, "Approximating dense cases of covering problems," in *Proc. DIMACS Workshop Netw. Design Connectivity Facilities Location*, 1998, vol. 40, pp. 169–178.

[6] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. 18th ACM Int. Conf. World Wide Web*, 2009, pp. 797–800.

[7] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. 27th IEEE Int. Conf. Data Eng.*, 2011, pp. 900–911.

[8] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2012, pp. 195–203.

[9] O. Berman and G. Y. Handler, "Optimal minimax path of a single service unit on a network to nonservice destinations," *Transp. Sci.*, vol. 21, no. 2, pp. 115–122, 1987.

[10] L. Zhang, J. Liu, H. Jiang, and Y. Guan, "Senstrack: Energy-efficient location tracking with smartphone sensors," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3775–3784, Oct. 2013.

[11] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3051–3063, Nov. 2015.

[12] G. Wang *et al.*, "Defending against sybil devices in crowdsourced mapping services," in *Proc. 14th ACM Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, Singapore, Jun. 2016.

[13] S. Lee, B. Kim, H. Kim, R. Ha, and H. Cha, "Inertial sensor-based indoor pedestrian localization with minimum 802.15. 4a configuration," *IEEE Trans. Ind. Inform.*, vol. 7, no. 3, pp. 455–466, Aug. 2011.

[14] O. J. Woodman, "An introduction to inertial navigation," Comput. Laboratory, Univ. of Cambridge, Cambridge, U.K., Tech. Rep. UCAMCL-TR-696, vol. 14, p. 15, 2007.

[15] L.-Y. Wei, W.-C. Peng, B.-C. Chen, and T.-W. Lin, "Pats: A framework of pattern-aware trajectory search," in *Proc. 11th IEEE Int. Conf. Mobile Data Manage.*, 2010, pp. 372–377.

[16] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemp. Phys.*, vol. 46, no. 5, pp. 323–351, 2005.

[17] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *Proc. 11th ACM Int. Conf. Extending Database Technol. Adv. Database Technol*, 2008, pp. 205–216.

[18] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie, "Searching trajectories by locations: An efficiency study," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 255–266.

[19] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2011, pp. 316–324.

[20] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 713–724.

[21] X. Fan, J. Liu, Z. Wang, and Y. Jiang, "Navigating the last mile with crowdsourced driving information," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2016, pp. 346–351.

[22] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu, "Crowdatlas: Self-updating maps for cloud and personal use," in *Proc. 11th ACM Annu. Int. Conf. Mobile Syst. Appl. Services*, 2013, pp. 469–470.

[23] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[24] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun, "Smartphones as practical and secure location verification tokens for payments," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2014.

[25] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proc. 10th ACM Workshop Mobile Comput. Syst. Appl.*, 2009, Art. no. 3.

Xiaoyi Fan (S'14) received the B.E. degree from Beijing University of Posts and Telecommunication, Beijing, China, in 2013, and the M.Sc. degree from Simon Fraser University, Burnaby, BC, Canada, in 2015. He is currently working toward the Ph.D. degree in the School of Computing Science, Simon Fraser University.

His areas of interest include cloud computing, big data, and mobile computing.

Jiangchuan Liu (S'01–M'03–SM'08) received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2003, both in computer science.

He is a University Professor in the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, and an NSERC E.W.R. Steacie Memorial Fellow. He is an EMC-Endowed Visiting Chair Professor of Tsinghua University, Beijing, China (2013–2016). From 2003 to 2004, he was an Assistant Professor at the Chinese University of Hong Kong.

Dr. Liu is a corecipient of the inaugural Test of Time Paper Award of IEEE INFOCOM (2015), ACM SIGMM TOMCCAP Nicolas D. Georganas Best Paper Award (2013), and ACM Multimedia Best Paper Award (2012).

Zhi Wang (S'10–M'14) received the B.E. and Ph.D. degrees in computer science in 2008 and 2014, from Tsinghua University, Beijing, China.

He is currently an Assistant Professor in the Graduate School at Shenzhen, Tsinghua University. His research areas include online social networks, mobile cloud computing and multimedia big data.

Dr. Wang received the China Computer Federation Outstanding Doctoral Dissertation Award (2014), ACM Multimedia Best Paper Award (2012), and MMM Best Student Paper Award (2015).

Yong Jiang received the Ph.D. degree from the Tsinghua University, Beijing, China, in 2002.

He is the Professor in the Information Sciences Department at the Graduate School at Shenzhen, Tsinghua University. He was an Assistant Professor (2002–2005) and Associate Professor (2005–2013) of information sciences at the Graduate School at Shenzhen, Tsinghua University. His research interests include computer network architecture to internet applications, with current foci on future internet architecture and big-data applications.

Xue (Steve) Liu received the Ph.D. degree in computer science (with multiple honors) from the University of Illinois at Urbana-Champaign, Champaign, IL, USA.

He is a William Dawson Scholar and Associate Professor in the School of Computer Science, McGill University, Montréal, QC, USA. He also was the Samuel R. Thompson Chaired Associate Professor in the University of Nebraska-Lincoln, and ~~as~~ a visiting faculty at HP Labs, Palo Alto, CA, USA. His research interests include computer and communication networks, real-time and embedded systems, cyber–physical systems and Internet of Things, green computing, and smart energy technologies. He has published more than 200 research papers in major peer-reviewed international journals and conference proceedings in these areas and received several best paper awards. His research has been covered by various news media reports~~**~~.