Itoi *et al. Scientific Phone Apps and Mobile Devices* (2017) 3:1
DOI 10.1186/s41070-017-0013-1

Scientific Phone Apps
and Mobile Devices

RESEARCH ARTICLE Open Access

# An offline mobile application for automatic evacuation guiding in outdoor environments

Junki Itoi, Masahiro Sasabe*, Jun Kawahara and Shoji Kasahara

**Abstract**

When a large-scale disaster occurs, evacuees have to move to safe places speedily, under highly stressful situations. For this purpose, an automatic evacuation guiding scheme based on unconscious cooperation between evacuees and their mobile nodes has been proposed and its performance, e.g., average/maximum evacuation times, has been well studied through simulation experiments. In this paper, we focus on the mobile application, which plays the key role of the automatic evacuation guiding scheme. The mobile application should be designed to operate even under offline environments because it may be disconnected from the network due to highly damaged communication infrastructures. We first design such an offline mobile application with required functions, i.e., positioning, estimation of evacuees' situations, process of road network data, and presentation of useful information. Since positioning data obtained by Global Positioning System (GPS) may include errors, we also propose a scheme to estimate the evacuees' situations, e.g., current position on road network and encounter with blocked road segments, by applying map matching algorithms, which try to identify a correct position and a road segment of an evacuee from erroneous positioning data. We further implement the offline application on an actual mobile phone. Through experiments, we evaluate the fundamental characteristics of the application, i.e., GPS positioning accuracy, processing time, and continuous operating time. Furthermore, we show the effectiveness of the proposed scheme in terms of estimation accuracy of an actual evacuation route, and that of blocked road segments.

**Keywords:** Automatic evacuation guiding in outdoor environments, Offline mobile application, Estimation of evacuees' situations, Map matching

## Background

In the 2011 Great East Japan Earthquake, many people could not quickly obtain important information using their own mobile phones (mobile nodes), due to damage to the communication infrastructures (Ministry of Internal Affairs and Communications). In addition, under such highly stressful situations, it may be difficult for disaster victims to operate their mobile nodes as usual. To tackle these problems, an automatic evacuation guiding scheme based on unconscious cooperation between evacuees and their mobile nodes has been proposed (Komatsu et al. 2016).

*Correspondence: sasabe@is.naist.jp
Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, 630-0192 Nara, Japan

This scheme is designed for outdoor environments. At first, each mobile node pre-installs a mobile application, which acquires a map and locations of safe places, e.g., primary evacuation areas and regional refuge areas, in usual time. When a disaster occurs, the mobile application first measures the location of its owner of evacuee by using existing measurement techniques, e.g., Global Positioning System (GPS). Then, it provides the evacuee with a recommended route, which starts from his/her location to the nearest safe place. Although the evacuee tries to follow the recommended route, he/she may encounter a blocked road segment on the route. In such a situation, the evacuee will select other action, e.g., proceeding to other road segment or entering a nearby building, etc., at his/her own decision. Note that the application can track this evacuee's evacuation route by periodically measuring his/her positions. As a result, it can automatically

estimate a blocked road segment based on the difference between recommended route and actual evacuation route. Then, it calculates and shows a new route that does not include the discovered blocked-road-segment. It can also share the discovered blocked-road-segments with other mobile applications and/or a cloud system, by using Delay Tolerant Network (DTN) (Fall 2003) and/or remaining communication infrastructures.

In (Komatsu et al. 2016), the authors focus on showing the fundamental characteristics of the automatic evacuation guiding scheme by demonstrating average/maximum evacuation times through several simulation experiments. In this paper, we focus on the mobile application, which plays the key role of the automatic evacuation guiding scheme. The mobile application should be designed to be able to operate even under offline environments because it may be disconnected from the network due to highly damaged communication infrastructures. We first design such an offline mobile application with required functions, i.e., positioning, estimation of evacuees' situations, process of road network data, and presentation of useful information.

Since the previous work does not consider the effects of GPS positioning errors, we also propose a scheme to estimate the evacuees' situations, i.e., traveling on road segments, traveling through junctions, and encounter with blocked road segments, from such erroneous positioning data. This is a kind of problems called map-matching (Quddus et al. 2007; Hashemi and Karimi 2014). Map-matching is an algorithm that matches positions including measurement errors to appropriate road segments.

According to the design, we develop a prototype of the offline mobile application. Our application is a hybrid application that can run over multiple platforms, e.g., iOS and Android, and consists of open-source software and open data. Through several experiments, we show the fundamental characteristics of our application, in terms of GPS positioning accuracy, processing time, and continuous operating time. We further evaluate the effectiveness of the proposed estimation scheme, in terms of estimation accuracy of an actual evacuation route, and that of blocked road segments.

The main contributions of this paper are as follows:

1. An offline mobile application for automatic evacuation guiding is designed and implemented as a hybrid application using existing open-source software and open data, which contributes to the penetration and further development of the application.
2. To enable automatic evacuation guiding, regardless of the spec of evacuees' mobile nodes, a lightweight scheme to estimate the evacuees' situations, e.g., current position on road network and encounter with blocked road segments, is proposed with the help of map matching algorithms.
3. Through several experiments, the fundamental characteristics of the proposed scheme is demonstrated, in terms of GPS positioning accuracy for moving objects, processing time and continuous operating time, estimation accuracy of traveling through junctions, and estimation accuracy of blocked road segment.

## Related work

The proliferation of mobile nodes, e.g., smart phones, has been opening a new vista of evacuation guiding (Iizuka et al. 2011; Winter et al. 2011; Komatsu et al. 2016). Iizuka et al. propose an evacuation guiding system using an ad hoc network whose connectivity is almost always guaranteed (Iizuka et al. 2011). It can present evacuees with both evacuation routes and timing to avoid crowds of evacuees. Winter et al. propose an evacuation guiding system using evacuees' trajectories (Winter et al. 2011). Evacuees continuously measure their trajectories by their mobile phones, share the trajectories with others through direct wireless communications, and try to find out available paths to safe places using the collected trajectories. Recently, Komatsu et al. propose a new concept of automatic evacuation guiding using evacuees' mobile phones, which can estimate the surrounding situations and guide evacuees without any operations of them (Komatsu et al. 2016). In this paper, we realize a mobile application for this automatic evacuation guiding.

As for outdoor positioning, GPS becomes one of the major positioning technologies in navigation systems, e.g., Intelligent Transport System (ITS) and Google Maps (Google). GPS, however, also causes gaps in positioning (Langley 1997). Map-matching algorithms integrate such erroneous positioning data with road network data such that they estimate the correct road segment on which the moving object, e.g., vehicle and pedestrian, is traveling, and the location on that road segment (Quddus et al. 2007; Hashemi and Karimi 2014).

Map-matching algorithms can be classified into four groups: geometric, topological, probabilistic, and advanced. Geometric map-matching algorithms simply use the geometric relationship, e.g., distance between positions and road segments (Greenfeld 2002; Bernstein and Kornhauser; Bentley and Maurer 1980). Topological map-matching algorithms consider the topological structure of road network besides the geometric relationship (Greenfeld 2002). Probabilistic map-matching algorithms define an elliptical or rectangular area as a confidence region, and match positions to road segments in the area (Honey et al.). Advanced map-matching algorithms use more refined techniques such

as a Kalman Filter (Krakiwsky et al. 1988) or a particle filter (Gustafsson et al. 2002).

Taking account of implementation simplicity and low computation complexity, which are essential features to run the application even over low-spec mobile nodes, we adopt the combination of geometric and topological approaches. Specifically, we use point-to-curve matching that matches GPS positions to their nearest road segments (Bernstein and Kornhauser). Note that the GPS positions can be sophisticated by applying Adaptive Kalman Filtering (Hu et al. 2009), which is more robust to the sudden changes of moving motion and measurement errors than the conventional Kalman Filtering. We also propose a topological map-matching algorithm to estimate the actual evacuation routes after traveling through junctions.

## Methods

### Design and implementation of mobile application for automatic evacuation guiding

#### *Overall structure of application*

Figure 1 shows the overall structure of mobile application for automatic evacuation guiding, where required functions and data, with their relationship, are described. First, evacuees pre-install the application into their own mobile nodes and obtain *map*, *locations of safe places*, and *graph*, i.e., road network, before disasters occur. After the application starts, it measures the current location by positioning, calculates a *recommended route* to the nearest safe place by route search, and presents the map and recommended route by presentation of useful information. Moreover, it periodically conducts positioning to grasp the evacuee's actual movement as *trajectory*, and automatically estimates *blocked road segments* from the difference between *recommended route* and *trajectory*, using estimation of evacuee's situation. If it newly discovers a blocked road segment, it recalculates *recommended route* by route search. Finally, it shares information about *blocked road segments* with other mobile nodes through direct wireless communication and/or with a cloud system through remaining communication infrastructures, by conducting communication control.

We develop the application as a hybrid application using Apache Cordova (Apache), so that the application can run over multiple platforms, e.g., iOS and Android. Since Apache Cordova is based on JavaScript, we also use some JavaScript-based open source software. In the following subsections, we give the detail of each function in Fig. 1.

#### *Positioning*

This function periodically measures the positions of mobile node using a GPS sensor, and stores the series of measured positions, i.e., trajectory, into the local database. We implement this function using Cordova Geolocation API (W3C). Geolocation API has two methods to measure positions. One is getCurrentPosition that measures positions at a fixed interval and the other is watchPosition that measures positions whenever the location changes. We adopt watchPosition because we observed that it is superior to getCurrentPosition in terms of positioning accuracy and power consumption, through preliminary experiments.

#### *Graph data handling*

The application needs a graph of the road network, where edges and nodes correspond to road segments

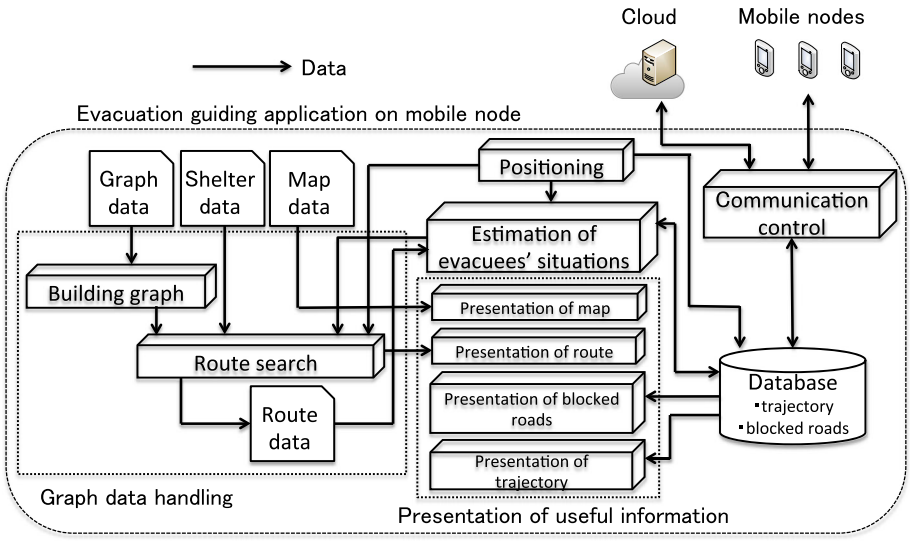


**Fig. 1** Overall structure of mobile application for automatic evacuation guiding

and endpoints of them, respectively. Since it should work even without any communication environments, it locally constructs the graph from XML-based OSM data retrieved from OpenStreetMap (OpenStreetMap). We use Cytoscape (Cytoscape) to handle graph data, e.g., calculating shortest paths as recommended routes. We should note here that OSM data for a target region may not be perfect because they are voluntarily managed by multiple users. For example, some road segments are identical but redundantly registered, and some intersections lose connections with the corresponding road segments. In this paper, we conducted pre-processing to the OSM data to fix the above problems.

#### *Presentation of useful information*

The application also obtains map tiles, which are fragments of images of the map, from OpenStreetMap. We use Leaflet (Leaflet), which is an open-source JavaScript library for mobile-friendly interactive maps, to display useful information, i.e., map, recommended routes, blocked road segments, and trajectory. To achieve real-time guiding, the application updates all the information except map whenever it obtains new information.

#### *Estimation of evacuees' situations*

The application estimates evacuees' situations based on map, recommended route, and trajectory, i.e., actual evacuation route. The evacuees' situations can be classified into three cases: traveling on the recommended route, traveling through junctions, and encountering blocked road segments. Since positioning errors are not considered in (Komatsu et al. 2016), we newly develop a scheme to estimate the evacuees' situations, with the help of map-matching algorithm. The detail of the estimation scheme will be given in the next section.

#### *Communication control*

In this paper, we focus on offline functions of the mobile application, which includes all the functions except communication control. In what follows, we briefly give the information about communication control. The application shares the information of blocked road segments with other mobile nodes and the server on cloud system. The communication with other nodes can be done through remaining communication infrastructures and/or direct wireless communication, e.g., Bluetooth and Wi-Fi Direct.

### Algorithm for estimation of evacuees' situations

#### *Overview*

The application periodically estimates evacuees' situations by taking account of real-time guiding, estimation accuracy, and power consumption of mobile nodes. Hereafter, we refer to the estimation scheme as the proposed scheme. As mentioned in the previous section, it continuously measures positions of the mobile node using watchPosition. When the application starts, it first obtains the initial location and immediately conducts the proposed scheme. At the succeeding positioning, it conducts the proposed scheme only when the time interval from the previous execution of the proposed scheme is not less than $I_{\mathrm{M}}$ ($I_{\mathrm{M}} > 0$). As a result, the application can conduct the proposed scheme at an approximate interval of $I_{\mathrm{M}}$. $I_{\mathrm{M}}$ should be at least longer than the processing time of the proposed scheme. In the next section, we will evaluate the impact of $I_{\mathrm{M}}$ on the system performance.

Algorithm 1 shows the estimation of evacuees' situations, where each mobile node's behavior at $i$-th ($i = 1, 2, \ldots$) round of the proposed scheme is described. Table 1 and Fig. 2 give the notation and geographical relationship among them, respectively. When the application starts at time $t(0)$, it first initializes $i = 0$ and set $p(0), \mathbf{r}, e(0), p_e(0), v_e(0)$ to be null, respectively. Algorithm 1 consists of three kinds of functions: (1) road-matching, i.e., estimation of the road segment on which the evacuee is traveling, (2) detection of leaving the road

**Algorithm 1** Estimation of evacuees' situations at $i$-th round

**Input:** $i, e(i-1), p_e(i-1), v_e(i-1), \mathbf{r}, G, \mathcal{E}_{\mathrm{NG}}$
**Output:** $e(i), p_e(i), v_e(i), \mathbf{r}, \mathcal{E}_{\mathrm{NG}}$

$(p(i), t(i)) :=$ getPos()
**if** $e(i-1) = null$ **then**
  $(\mathbf{r}, e(i), p_e(i), v_e(i)) :=$ getRoute$(p(i), v_{\mathrm{d}}, G, \mathcal{E}_{\mathrm{NG}})$
  **return** $(e(i), p_e(i), v_e(i), \mathbf{r}, \mathcal{E}_{\mathrm{NG}})$
**end if**
$(e(i), p_e(i), v_e(i)) :=$ getNearestEdge$(p(i), \mathbf{r})$
**if** $e(i) = e(i-1)$ **then**
  **if** $d_{\mathrm{E}}(p(i), p_e(i)) > \theta_{\mathrm{D}}$ **or** $\frac{d_{\mathrm{G}}(p_e(i), p_e(i-1))}{t(i)-t(i-1)} < \theta_{\mathrm{V}}$ **or** $d_{\mathrm{G}}(p_e(i), v_e(i)) > d_{\mathrm{G}}(p_e(i-1), v_e(i-1))$ **then**
    $\mathcal{E}_{\mathrm{NG}} := \mathcal{E}_{\mathrm{NG}} \cup$ getNextEdge$(e(i), r)$
    $(\mathbf{r}, e(i), p_e(i), v_e(i)) :=$ getRoute$(p(i), v_{\mathrm{d}}, G, \mathcal{E}_{\mathrm{NG}})$
  **end if**
**else**
  $\mathcal{E}_{\mathrm{C}} :=$ getConnectedEdges$(v_e(i-1), G, \mathcal{E}_{\mathrm{NG}})$
  **for** $e' \in \mathcal{E}_{\mathrm{C}}$ **do**
    $(e', p_{e'}, v_{e'}) :=$ getNearestEdge$(p(i), (e'))$
  **end for**
  $\widehat{e}(i) := \arg\max_{e' \in \mathcal{E}_{\mathrm{C}}} \left( \frac{d_{\mathrm{G}}(p_{e'}, p_e(i-1))}{t(i)-t(i-1)} \right)$
  **if** $\widehat{e}(i) \neq e(i)$ **then**
    $\mathcal{E}_{\mathrm{NG}} := \mathcal{E}_{\mathrm{NG}} \cup \{e(i)\}$
    $(\mathbf{r}, e(i), p_e(i), v_e(i)) :=$ getRoute$(p(i), v_{\mathrm{d}}, G, \mathcal{E}_{\mathrm{NG}})$
  **end if**
**end if**
**return** $(e(i), p_e(i), v_e(i), \mathbf{r}, \mathcal{E}_{\mathrm{NG}})$

**Table 1** Notation

| Notation | Definition |
|---|---|
| $t(i)$ | Start time of $i$-th round |
| $p(i)$ | Measured position at $i$-th round |
| $r$ | Recommended route, i.e., sequence of edges |
| $e(i)$ | The nearest edge in recommended route from $p(i)$ |
| $p_e(i)$ | Estimated position on $e(i)$ |
| $v_e(i)$ | $e(i)$'s endpoint in traveling direction |
| $v_\mathrm{d}$ | Destination node |
| $G$ | Graph composed of nodes and edges |
| $\mathcal{E}_\mathrm{NG}$ | Set of blocked road segments |

segment on which the evacuee traveled, and (3) estimation of traveling through a junction. With the help of these functions, the application updates the information, i.e., the recommended route, the evacuee's traveling road segment, and the set of blocked road segments. In what follows, we will give the detail of each function.

#### *Road-matching*

Road-matching is given by lines 2–6 in Algorithm 1. If the evacuee did not travel on any edge at $(i-1)$-th round, e.g., initial state, the algorithm tries to calculate recommended route $\mathbf{r}$ and the nearest edge $e(i)$ by using $\mathsf{getRoute}(p(i), v_\mathrm{d}, G, \mathcal{E}_\mathrm{NG})$. Otherwise, it updates the nearest edge $e(i)$ by using $\mathsf{getNearestEdge}(p(i), \mathbf{r})$. In what follows, we give the detail of these functions.

We achieve getNearestEdge() using point-to-curve matching (Bernstein and Kornhauser), which matches an erroneous position to its nearest road segment by calculating the distance from the measured position to each candidate road segment. Given position $p$ and candidate set $\mathbf{e}$ of edges, $\mathsf{getNearestEdge}(p, \mathbf{e})$ returns the nearest edge $e = \arg\min_{e' \in \mathbf{e}} d_\mathrm{E}(p, e')$, the nearest position $p_e$ on $e$ from $p$, and endpoint $v_e$ of $e$ in the traveling direction. Here, $d_\mathrm{E}(p, e)$ calculates the distance between edge $e$ and position $p$, which is equivalent to the distance between $p_e$ and $p$.

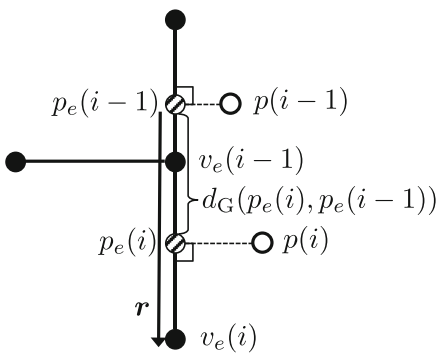


**Fig. 2** Geographical relationship among notation

$\mathsf{getRoute}(p, v_\mathrm{d}, G, \mathcal{E}_\mathrm{NG})$ calculates the recommended route, e.g., the shortest path, from $p$ to $v_\mathrm{d}$. Given position $p$, destination node $v_\mathrm{d}$, graph $G$, and set $\mathcal{E}_\mathrm{NG}$ of blocked road segments, it first calculates the nearest edge $e$ and point $p_e$ on $e$ by using $\mathsf{getNearestEdge}(p, \mathcal{E})$ where $\mathcal{E}$ is the set of all edges in $G$. Second, it searches for recommended route $\mathbf{r}$ from $p_e$ to $v_\mathrm{d}$, which does not include any edge in $\mathcal{E}_\mathrm{NG}$. $\mathsf{getRoute}(p, v_\mathrm{d}, G, \mathcal{E}_\mathrm{NG})$ returns recommended route $\mathbf{r}$, the nearest edge $e$, the nearest point $p_e$ on $e$, and end point $v_e$ on $e$ in the traveling direction. It, however, returns null for each variable if the current position $p$ is apart from the nearest edge $e$ at a certain distance, i.e., $d_\mathrm{E}(p, p_e) > \theta_\mathrm{D}$ ($\theta_\mathrm{D} > 0$). Since appropriate value of $\theta_\mathrm{D}$ depends on the positioning accuracy, we will discuss this point in the next section.

#### *Detection of leaving road segment*

Detection of leaving the road segment on which the evacuee traveled is given by lines 7–11 in Algorithm 1. If $i$-th round's nearest edge $e(i)$ equals to previous one $e(i-1)$, the proposed algorithm basically estimates that the evacuee smoothly follows recommended route $\mathbf{r}$. It, however, detects leaving edge $e(i-1)$ if one of the following conditions is satisfied:

- Distance $d_\mathrm{E}(p(i), p_e(i))$ between current position $p(i)$ and the nearest point $p_e(i)$ on $e(i)$ is larger than threshold $\theta_\mathrm{D}$ of distance. This case indicates that the evacuee is not traveling on any road segment but is in free space, e.g., park, building, etc. $\theta_\mathrm{D}$ should be large enough to eliminate the effect of positioning errors. Note that GPS positioning accuracy will degrade when evacuees enter buildings. To improve the positioning accuracy in indoor environments, which is out of our scope, our system can cooperate with existing indoor positioning systems (Liu et al. 2007). Some of such indoor positioning systems can also be GPS-based, such as wireless assisted GPS (AGPS) and GPS weak signal tracking technology, e.g., SuperSense.
- Average traveling speed $\frac{d_\mathrm{G}(p_e(i), p_e(i-1))}{t(i)-t(i-1)}$ at the interval between $(i-1)$-th and $i$-th rounds is less than threshold $\theta_\mathrm{V}$ of traveling speed. Note that $d_\mathrm{G}(p_e(i), p_e(i-1))$ gives the distance between $p_e(i)$ and $p_e(i-1)$ along edges of the graph. This case will occur when the evacuee leaves current recommended route $\mathbf{r}$, as shown in Fig. 3a.
- Traveling direction at $i$-th round is opposite to that at $(i-1)$-th round. This case will occur when the

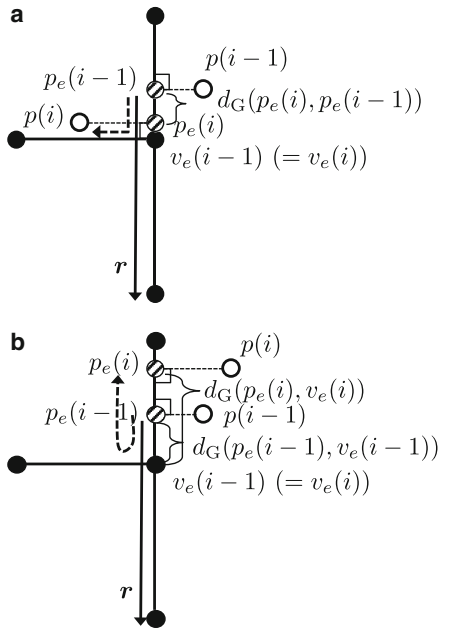


**Fig. 3** Detection of leaving road segment. **a** Slowdown of average traveling speed. **b** Reversal of traveling direction

evacuee goes back the road segment. The condition can be expressed by $d_G(p_e(i), v_e(i)) > d_G(p_e(i-1), v_e(i-1))$, as shown in Fig. 3b.

If one of the above conditions is satisfied, the algorithm estimates that next road segment $\mathsf{getNextEdge}(e(i), \mathbf{r})$ of $e(i)$ on $\mathbf{r}$ may be blocked and adds it to set $\mathcal{E}_{NG}$ of blocked road segments. Finally, the algorithm recalculates a new recommended route and the nearest edge on the route by using $\mathsf{getRoute}(p, v_d, G, \mathcal{E}_{NG})$.

#### *Estimation of traveling through junction*

Estimation of traveling through junction is given by lines 12–22 in Algorithm 1. If edge $e(i)$ at $i$-th round is not equal to edge $e(i-1)$ at $(i-1)$-th round, the algorithm estimates that the evacuee travels through a junction.

The algorithm first obtains set $\mathcal{E}_C$ of candidate road segments on which the evacuee may be traveling, by using $\mathsf{getConnectedEdges}(v_e(i-1), G, \mathcal{E}_{NG})$. Since the evacuee was traveling from $p_e(i-1)$ to $v_e(i-1)$ on edge $e(i-1)$ at $(i-1)$-th round, we define $\mathcal{E}_C$ as the set of road segments that are directly connected to $v_e(i-1)$. Next, the algorithm calculates the average traveling speed for each candidate edge $e' \in \mathcal{E}_C$ under the assumption that the evacuee travels edge $e'$. Since it can be expected that the average traveling speed for the edge on which the evacuee is actually traveling becomes faster than those for other candidate edges, the algorithm selects the edge with the fastest traveling speed as $\widehat{e}(i)$. If $\widehat{e}(i)$ equals to $e(i)$, the algorithm estimates the evacuee is traveling on $e(i)$ as expected. Otherwise, it estimates edge $e(i)$ is blocked and recalculates a recommended route.

## Results and discussion

We conducted several experiments using actual mobile phones, i.e., Nexus 5 with Android OS 4.4.2. Figure 4 illustrates a sample screenshot of our application. Through preliminary experiments, we first clarify the fundamental characteristics of our application in terms of positioning accuracy, processing time, and continuous operating time. Next, we show the estimation accuracy of proposed scheme.


**Fig. 4** Screenshot of our application, which displays recommended route (*red line*), original trajectory (*blue line*), estimated evacuation route (*black line*), and blocked road segment (*red cross*)

### Evaluation scenario

We conducted fifty independent experiments in the campus of our institute. Figure 5 shows the experimental scenario as a graph. There are eleven road segments labeled $e_1$ to $e_{11}$ and one blocked road segment $e_5$. In each experiment, an evacuee with Nexus 5 starts his evacuation from node $v_s$. At the same time, our application obtains the initial location and finds out the nearest safe place $v_d$. Then, it calculates first recommended route $\mathbf{r}_1 = (e_1, e_5, e_6, e_7)$. The evacuee goes along $\mathbf{r}_1$ but encounters blocked road segment $e_5$ at first junction $v_1$. He changes the route by himself and proceeds to $e_2$ instead of $e_5$. The application detects that the evacuee leaves $\mathbf{r}_1$ using the proposed scheme and adds $e_5$ to $\mathcal{E}_{NG}$. It also calculates new route $\mathbf{r}_2 = (e_2, e_3, e_4)$, which does not include $e_5$. Finally, the evacuee can smoothly reach destination $v_d$ along $\mathbf{r}_2$.

In the above scenario, the correct evacuation route should be $\mathbf{t}_{true} = (e_1, e_2, e_3, e_4)$. On the contrary, the application obtains estimated evacuation route $\mathbf{t}_{est}$ at each experiment. Since it periodically estimates the road segment on which the evacuee may be traveling, the estimated trajectory may include two or more consecutively identical edges. We obtain $\mathbf{t}_{est}$ by eliminating such redundant information from the estimated trajectory. Note that there are three junctions $v_1, v_2, v_3$ in $\mathbf{t}_{true}$ and they have correct next edges, $e_2, e_3, e_4$, respectively. We define successful probability $T_{true}(v)$ ($v \in \{v_1, v_2, v_3\}$), which is the ratio of the number of experiments where the proposed scheme can correctly estimate the next edge of $v$ to the total number of experiments. We also define $\overline{T}_{true}$ as the average of $T_{true}(v)$ among $v_1, v_2$, and $v_3$.

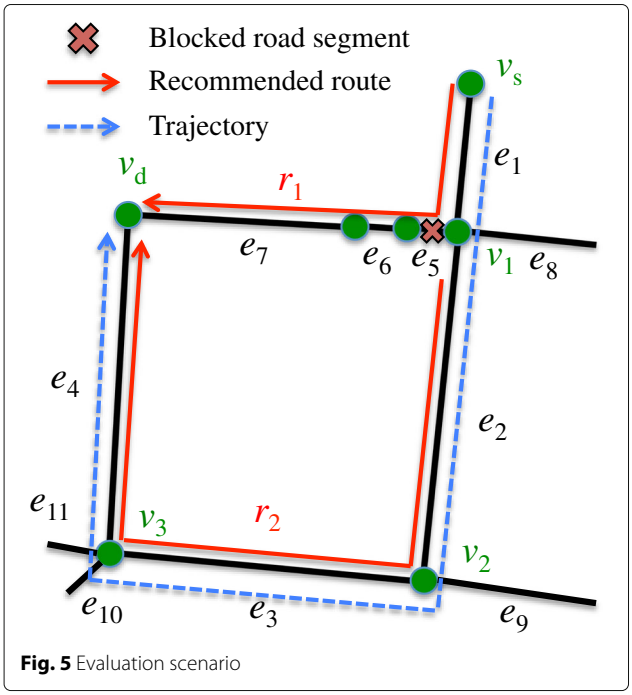


**Fig. 5** Evaluation scenario

To evaluate the estimation accuracy of blocked road segment, we apply precision $B_{precision}$, recall $B_{recall}$, and F-measure $B_{F-measure}$ (Christopher et al. 2008). Precision means how correct the estimation results are while recall means how complete the estimation results are. F-measure is the harmonic mean of precision and recall. Specifically, these criteria are defined as follows: $B_{precision} = \frac{B_{TP}}{B_{TP}+B_{FP}}$, $B_{recall} = \frac{B_{TP}}{B_{TP}+B_{FN}}$, and $B_{F-measure} = \frac{2B_{recall} \times B_{precision}}{B_{recall}+B_{precision}}$, where $B_{TP}$ is the number of correctly estimated blocked-road-segments, $B_{FP}$ is the number of incorrectly estimated blocked-road-segments, and $B_{FN}$ is the number of undetected blocked-road-segments.

During the experiments, the evacuee tried to move at a constant speed and the average traveling speed was 1.25 [m/s]. We set threshold $\theta_V$ of traveling speed to detect leaving a road segment to be 0.5 [m/s], which is less than half of the average traveling speed.

### GPS positioning accuracy for moving object

There have been several reports about GPS positioning errors. In (Committee), it is assumed that they follow a normal distribution. On the contrary, Zandbergen shows that they follow a Rayleigh distribution through experiments where the positioning data are collected by a GPS unit located at a fixed point. In our scenario, we have to reveal the GPS positioning accuracy for moving objects, but it is a challenging problem. Figure 6 illustrates current (actual) edge $e(i)$, measured position $p(i)$, estimated position $p_e(i)$, and actual position $p_e^{true}(i)$ of mobile node at $i$-th round in Algorithm 1. It is impossible to obtain actual position $p_e^{true}(i)$ for measured position $p(i)$ in actual situations. In addition, point-to-curve matching focuses on the distance between $p(i)$ and $p_e(i)$ rather than that between $p(i)$ and $p_e^{true}(i)$. Taking account these characteristics, we define

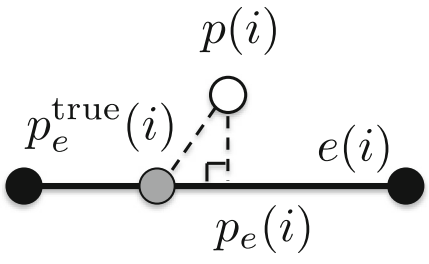


**Fig. 6** Current (actual) edge $e(i)$, measured position $p(i)$, estimated position $p_e(i)$, and actual position $p_e^{true}(i)$ of mobile node at $i$-th round

the absolute positioning error at $i$-th round as distance $d_{\mathrm{E}}(p(i), p_e(i))$.

We obtained 1252 samples of positioning data through an experiment where the evacuee traveled back and forth a road segment in our campus. Figure 7 illustrates the histogram of absolute positioning errors, where we omit the direction of gaps. We also show a normal distribution whose average is 0 [m] and standard deviation $\sigma$ is 6.83 [m]. We observe that the GPS positioning errors can be roughly approximated by the normal distribution. Taking account of the characteristics of a normal distribution, i.e., 95.45% of positioning data fall in the range of $0 \pm 2\sigma$, we set $\theta_{\mathrm{D}}$ to be $2\sigma = 3.66\ [m]$.

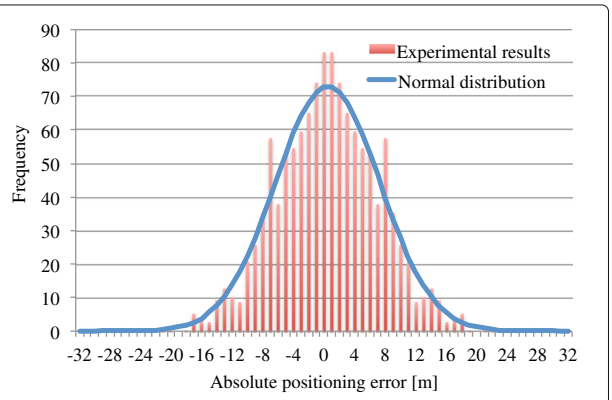


**Fig. 7** Histogram of GPS positioning errors

### Processing time and continuous operating time

Next, we focus on processing time and continuous operating time. Since the mobile application has to conduct offline route search, both both processing time and battery consumption increase with the size of road network. On the contrary, the size of road network should be large enough to include appropriate safe place(s). In Japan, there is a guideline where it is desirable to place a regional refuge area within a range of 2 [km] (Masuda 2002). In the experiments, we used the road network data with 4957 nodes, which corresponds to about one tenth of Ikoma city, i.e., 3.5 [km] × 3.0 [km], in Japan. We observed that the processing time of route search is less than 0.5 [s], which is small enough to achieve real-time guiding.

Figure 8 illustrates average continuous operating time for three cases: 1) *display*, 2) *positioning*, and 3) *reroute*. In *display* case, the application only displays the screen with the maximum brightness. In *positioning* case, it displays the screen with the maximum brightness and measure positions every 5 [s]. In *reroute* case, it displays the screen with the maximum brightness, measure positions and calculates routes every 5 [s]. We observe that our application can keep running about five hours even under the highly loaded situation, i.e., *reroute* case. In actual situations,

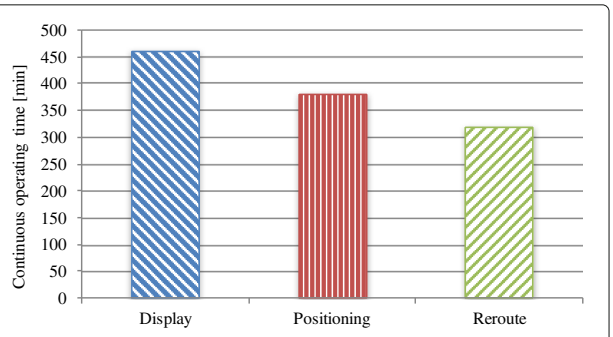


**Fig. 8** Average continuous operating time

the continuous operating time will be longer because our application conducts the route search only when it detects and/or retrieves blocked road segments.

Taking account of the above results, we set $I_{\mathrm{M}}$ to be 5, 10, and 15 [s]. In what follows, we show the average of fifty independent experiments for each $I_{\mathrm{M}}$.

### Estimation accuracy of traveling through junctions

Figure 9 shows successful probability $T_{\mathrm{true}}(v)$ of estimation of the next edge of junction $v$ ($v \in \{v_1, v_2, v_3\}$) and average $\overline{T}_{\mathrm{true}}$ among the junctions for $I_{\mathrm{M}} = 5, 10, 15$ [s]. We observe that $T_{\mathrm{true}}(v)$ monotonically increases with $I_{\mathrm{M}}$, regardless of $v$. This can be analyzed as follows. Recall the process of estimation of traveling through junction in the previous section. When the algorithm calculates the average traveling speed for each candidate edge $e' \in \mathcal{E}_{\mathrm{C}}$ (line 17 in Algorithm 1), $d_{\mathrm{G}}(p_{e'}, p_e(i-1))$ can be divided into $d_{\mathrm{G}}(p_{e'}, v_e(i-1))$ and $d_{\mathrm{G}}(v_e(i-1), p_e(i-1))$. Since $d_{\mathrm{G}}(v_e(i-1), p_e(i-1))$ is identical for each $e' \in \mathcal{E}_{\mathrm{C}}$, $d_{\mathrm{G}}(p_{e'}, v_e(i-1))$ only affects the estimation accuracy. When $I_{\mathrm{M}}$ becomes long, $d_{\mathrm{G}}(p_{e'}, v_e(i-1))$ only for the correct edge tends to increase, and thus the estimation

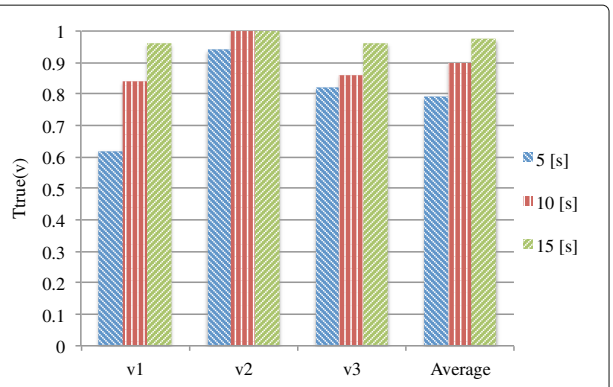


**Fig. 9** Successful probability $T_{\mathrm{true}}(v)$ of estimation of the next edge of junction $v$ ($v \in \{v_1, v_2, v_3\}$) and average $\overline{T}_{\mathrm{true}}$ among the junctions for $I_{\mathrm{M}} = 5, 10, 15$ [s]

accuracy increases. As a result, the proposed scheme can achieve about 98% of $\overline{T}_{\text{true}}$ in case of $I_{\text{M}} = 15$.

Next, we focus on relationship among successful probabilities of three junctions, $T_{\text{true}}(v_2) > T_{\text{true}}(v_3) > T_{\text{true}}(v_1)$. The following two factors influence successful probability $T_{\text{true}}(v)$.

- The number $N(v)$ of edges connected to junction $v$, i.e., degree.
- Existence of blocked road segment(s) connected to junction $v$.

The estimation becomes simple for a junction with small degree because the number of candidate edges also becomes small. As a result, estimation at $v_2$, which has the smallest degree, achieves the highest estimation accuracy. On the other hand, both $v_1$ and $v_3$ have the same degree of 4. $v_1$, however, has blocked road segment $e_5$ on its next edge. Thus, estimation at $v_1$ becomes more difficult than that at $v_3$.

### Estimation accuracy of blocked road segment

Table 2 gives the estimation accuracy of blocked road segment for $I_{\text{M}} = 5, 10, 15$. We observe that both $B_{\text{precision}}$ and $B_{\text{recall}}$ monotonically increase with $I_{\text{M}}$, and so does $B_{\text{F-measure}}$. Note that the estimation accuracy of blocked road segment $e_5$ is linked with that of the next edge of junction $v_1$, which is already given in the previous subsection. If the algorithm incorrectly conjectures that the evacuee moves to edge $e_5$ through junction $v_1$, it judges the next edge $e_6$ as a blocked road segment. In this case, both $B_{\text{FN}}$ and $B_{\text{FP}}$ increase by one, respectively. Furthermore, if incorrect estimation of the next edge at junctions $v_2$ and/or $v_3$ occur, it increases $B_{\text{FP}}$. As a result, $B_{\text{precision}}$ tends to be smaller than $B_{\text{recall}}$. We, however, confirm that the proposed scheme can achieve 0.95 of $B_{\text{F-measure}}$ for $I_{\text{M}} = 15$.

From the above results, we find that $I_{\text{M}} = 15$ is enough to achieve highly accurate estimation of both traveling through junctions and blocked road segment.

## Conclusion

In this paper, we designed and implemented an offline mobile application for automatic evacuation guiding based on cooperation between evacuees and their mobile nodes. Since the positioning data of mobile nodes are erroneous, we further proposed a scheme to estimate evacuees' situations with the help of map-matching algorithm. The proposed scheme consists of three functions: (1) road-matching, (2) detection of leaving road segment, and (3) estimation of traveling through junction. Through several experiments using actual mobile phones, we showed that our application with an appropriate control interval, e.g., $I_{\text{M}} = 15$, can correctly guide evacuees in a real-time manner and during a sufficiently long period, which is more than five hours.

As part of future work, combining the sensing results of GPS sensor with those of other types of sensors, e.g., gyroscopes and acceleration sensor, will help to further reduce $I_{\text{M}}$ while keeping the estimation accuracy. We also plan to implement the function of communication control and evaluate its practicality.

**Table 2** Estimation accuracy of blocked road segments

| $I_{\text{M}}$ [s] | 5 | 10 | 15 |
|---|---|---|---|
| $B_{\text{precision}}$ | 0.55 | 0.78 | 0.94 |
| $B_{\text{recall}}$ | 0.68 | 0.86 | 0.96 |
| $B_{\text{F-measure}}$ | 0.58 | 0.81 | 0.95 |

#### Authors' contributions

The authors have contributed to this paper and all authors read and approved the final manuscript.

#### Competing interests

The authors declare that they have no competing interests.

Received: 7 October 2016 Accepted: 17 January 2017
Published online: 26 January 2017

#### References

Apache. Apache Cordova. https://cordova.apache.org/. Accessed 22 Jan 2017.
Bernstein D, Kornhauser A. An Introduction to Map Matching for Personal Navigation Assistants. Technical report, Transportation Research Board 1998, https://trid.trb.org/view.aspx?id=681286.
Bentley JL, Maurer HA. Efficient worst-case data structures for range searching. Acta Informatica. 1980;13(2):155–68.
Christopher DM, Prabhakar R, Hinrich S. Introduction to Information Retrieval. New York: Cambridge University Press; 2008.
Committee FGD. Geospatial Positioning Accuracy Standards Part 3: National Standard for Spatial Data Accuracy. Technical report, Transportation Research Board 1998, https://www.fgdc.gov/standards/projects/accuracy/part3/index_html.
Cytoscape. Cytoscape. http://js.cytoscape.org/.
Fall K. A Delay-Tolerant Network Architecture for Challenged Internets. In: Proc. of SIGCOMM'03. New York: ACM; 2003. p. 27–34.
Google. Google Maps. https://maps.google.com. Accessed 22 Jan 2017.
Greenfeld JS. Matching GPS Observations to Locations on a Digital Map. In: Proc. of Transportation Research Board 81st Annual Meeting; 2002. https://trid.trb.org/view.aspx?id=711583.
Gustafsson F, Gunnarsson F, Bergman N, Forssell U, Jansson J, Karlsson R, Nordlund PJ. Particle filters for positioning, navigation, and tracking. IEEE Trans Signal Process. 2002;50(2):425–37.
Hashemi M, Karimi HA. A critical review of real-time map-matching algorithms: current issues and future directions. Comput Environ Urban Syst. 2014;48:153–65.
Honey SK, Zavoli WB, Milnes KA, Phillips AC, White Jr MS, Loughmiller Jr GE. Vehicle Navigational System and Method. Google Patents. US Patent 4,796,191.1989, https://www.google.com/patents/US5862511.
Hu C, Chen W, Chen Y, Liu D. Adaptive Kalman Filtering for Vehicle Navigation. J Global Positioning Syst Positioning. 2009;2(1):42–7.
Iizuka Y, Yoshida K, Iizuka K. An Effective Disaster Evacuation Assist System Utilized by an Ad-Hoc Network. In: Proc. of HCI International 2011. Berlin, Heidelberg: Springer; 2011. p. 31–5.
Komatsu N, Sasabe M, Kawahara J, Kasahara S. Automatic evacuation guiding scheme based on implicit interactions between evacuees and their mobile nodes. GeoInformatica. 20161–15. http://link.springer.com/article/10.1007/s10707-016-0270-1.

Krakiwsky EJ, Harris CB, Wong RV. A Kalman Filter for Integrating Dead Reckoning, Map Matching and GPS Positioning. In: Proc. of IEEE PLANS'88; 1988. p. 39–46. http://ieeexplore.ieee.org/document/195464/references.
Langley RB. The GPS Error Budget. GPS World. 1997;8(3):51–6.
Leaflet. Leaflet. http://leafletjs.com/. Accessed 22 Jan 2017.
Liu H, Darabi H, Banerjee P, Liu J. Survey of Wireless Indoor PositioningTechniques and Systems. IEEE Trans Syst Man Cybernet Part C. 2007;37(6):1067–1080.
Masuda N. Urban Park Planning from the Point of View of the Safe and Secure Urban Environment (in Japanese. J Japan Inst Landsc Archit. 2002;66(3): 180–4.
Ministry of Internal Affairs and Communications. 2011 WHITE PAPER Information and Communications in Japan. http://www.soumu.go.jp/johotsusintokei/whitepaper/eng/WP2011/2011-index.html. Accessed 22 Jan 2017.
OpenStreetMap. OpenStreetMap. https://www.openstreetmap.org/. Accessed 22 Jan 2017.
Quddus MA, Ochieng WY, Noland RB. Current map-matching algorithms for transport applications: state-of-the art and future research directions. Transp Res Part C Emerg Technol. 2007;15(5):312–28.
W3C. Geolocation API Specification W3C Recommendation 24 October 2013. http://www.w3.org/TR/geolocation-API/. Accessed 22 Jan 2017.
Winter S, Richter KF, Shi M, Gan HS. Get Me out of Here: Collaborative Evacuation Based on Local Knowledge. In: Proc. of Third International Workshop on Indoor Spatial Awareness. New York: ACM; 2011. p. 35–42.