Electronic Preprint for Journal of Information Processing Vol.29

Regular Paper

A Route Recommendation Method Considering Individual User's Preferences by Monte-Carlo Tree Search and Its Evaluations

Yuta Ishizaki[1,a)] Yurie Koyama[1,b)] Toshinori Takayama[2] Nozomu Togawa[1,c)]

Received: April 2, 2020, Accepted: October 6, 2020

Abstract: As smartphones and tablets are widely spread and used, route recommendation and guidance services have become commonplace. Conventional services in route recommendation and guidance try to give best routes in terms of route length, time required, and train/bus fares, whereas even different users are given the same route when inputting the same parameters. However, each user has various preferences from the aspect of safety and comfort. It is strongly desirable to reflect the user's preferences in route recommendation and recommend the most preferable route to every user. Since user's preferences are extremely vague and complicated, how to evaluate them in route recommendation is one of the key problems there. In this paper, we propose a route recommendation method, called *P-UCT method*, considering individual user's preferences utilizing Monte-Carlo tree search. In the proposed method, we firstly extract route features based on the route recommendation history of every user and construct a route evaluator based on Support Vector Machine (SVM). After that, the method generates a random route from a start point to an end point by Monte-Carlo tree search. The route evaluator determines how well every generated route matches the user's preferences. By repeating the evaluation, the method obtains the route, which must be closest to the user's preferences. Experimental results demonstrate that the proposed method outperforms the existing method from the viewpoint of the average evaluation scores. They also demonstrate that the proposed method provides the recommended route reflecting the user's individual preferences even if it learns the recommended route history of areas in different situations.

Keywords: Monte-Carlo tree search, UCT, SVM, route recommendation

1. Introduction

As smartphones and tablets are widely spread and used, route recommendation and guidance services have become commonplace. Conventional services in route recommendation and guidance try to give best routes in terms of route length, time required, and train/bus fares, whereas even different users are given the same route when inputting the same parameters [1], [2].

Generally, individual users have various preferences from the aspect of safety and comfort. In fact, according to the questionnaires in Ref. [3], it is shown that users prefer a route with sidewalks, a small number of turns, and a small number of slopes. It is strongly desirable to reflect the user's preferences in route recommendation and recommend the most preferable route to every user. However, since user's preferences are extremely vague and complicated, how to evaluate them in route recommendation becomes a great concern.

1.1 Previous Works

Route recommendation methods considering individual user's preferences proposed so far can be roughly classified into (1)–(3).

[1] Dept. of Computer Science and Communications Engineering, Waseda University, Shinjuku, Tokyo 169–8555, Japan
[2] Zenrin DataCom Co., LTD., Minato, Tokyo 108–0023, Japan
[a)] yuta.Ishizaki@togawa.cs.waseda.ac.jp
[b)] yurie.koyama@togawa.cs.waseda.ac.jp
[c)] togawa@togawa.cs.waseda.ac.jp

(1) A route recommendation method based on the Dijkstra method [3]
In Ref. [3], a method reflecting the user's preferences by applying the Dijkstra method is proposed. In this method, based on the questionnaire survey, the user's preferences are quantified and reflected in the weighting of the route cost and then the best possible route is recommended to every user.

(2) Route recommendation methods based on the genetic algorithm [4], [5], [6], [7]
The methods using the genetic algorithm (GA) encode route information as a gene. After that, GA generates a route by repeating the crossover and mutation of superior individuals calculated by a certain evaluation function. The advantages of using GA are that multi-purpose optimization can be performed simultaneously by designing a multi-purpose evaluation function and that a suboptimal solution can be derived. For example, in Ref. [4], a method using the multi-objective genetic algorithm (MOGA) for car navigation is proposed. In this method, each route is encoded into a gene and the best possible route is recommended to each user by repeating crossover and mutation.

(3) Route recommendation methods based on fuzzy measures and fuzzy integral [8], [9]
The methods using fuzzy measures and fuzzy integral evaluate a generated route by the non-additivity of fuzzy measure. The weight of the user's individual preferences is calculated

by taking into account the interaction such as the synergistic effect and the offset effect between the attributes that evaluate the route. The preference correlation is calculated non-additively, and the evaluation function is calculated using a weighted sum. For example, in Ref. [8], a route recommendation system based on the fuzzy measures and integral model is proposed. Based on the detailed questionnaire surveys for an individual user, the road weights are fully customized somehow for the user and the system recommends the satisfactory route.

In all of these previous researches, a user-specific route recommendation is realized by giving the weights to the route elements such as route length, number of turns, number of slopes based on the questionnaire surveys. However, even if individual user's preferences are reflected in individual route elements, the entire route may not match the user's preferences. For example, when a user prefers "a route in which the total length of stairs is equal to the total length of slopes", it is almost impossible to search for such a route, just considering the weights of the route elements. In this case, we have to evaluate an entire route, not an individual route element. Furthermore, in Ref. [8], customizing the road weights based on detailed questionnaire surveys for every user is necessary. However, it is generally too difficult for an individual user to customize the road weights to always give best preferable routes. In order to recommend a route that meets the user's preferences, it is essential to have a mechanism that evaluates an entire route without explicitly customizing the weights of route elements.

Now we focus on a Monte-Carlo tree search method [10], [11], [12] to tackle this problem. In Monte-Carlo tree search, we can evaluate an entire route by using the Monte-Carlo simulation. The key point in the success of Monte-Carlo tree search is how to set up the node evaluation index, called a *UCB1* (*Upper Confidence Bound 1*) value. In Refs. [10], [11], [12], the UCB1 value is set up by giving the appropriate reward to the Monte-Carlo simulation results. However, as far as we know, there exists no research in which Monte-Carlo tree search is applied to route search, nor effective design method for UCB1 value.

1.2 Our Proposal

In this paper, we propose a route recommendation method, called *P-UCT* (*Personalized Upper Confidence Tree*) *method*, considering individual user's preferences utilizing Monte-Carlo tree search [*1]. Particularly, we propose a *P-UCB1* (*Personalized Upper Confidence Bound 1*) *value* for node evaluation index so that the Monte-Carlo tree search is efficiently applied to route search. The proposed P-UCB1 value can be given to every intersection node in the given road network, which demonstrates how much the intersection node can be preferable for an individual user.

In the proposed method, we firstly extract route features based on the route recommendation history of every user and construct a route evaluator based on Support Vector Machine (SVM). By introducing SVM, we can evaluate an entire route without explicitly customizing the weights of route elements for every user. After that, the method generates a random route from a start point to an end point by Monte-Carlo tree search using P-UCB1 values. The route evaluator determines how well every generated route matches the user's preferences. By repeating the evaluation, the method obtains the route, which must be closest to the user's preferences.

1.3 Contributions of the Paper

The contributions of this paper are summarized as follows:

(1) We propose a route recommendation method utilizing Monte Carlo tree search, where we can evaluate an entire route and hence effectively realize route recommendation considering user's preferences.

(2) In order to apply Monte Carlo tree search to route recommendation, we newly propose a P-UCB1 value, which demonstrates how much the intersection node in a given road network can be preferable for an individual user.

(3) The experimental results demonstrate that the average score of the proposed method becomes 3.00 (the full score is 4.00) when 30 routes are learned beforehand, while that of the existing method becomes 1.875. We also demonstrate that the proposed method provides the recommended route reflecting the user's individual preferences even if it learns the recommended route history of areas in different situations.

1.4 Organization of the Paper

The rest of this paper is organized as follows: Section 2 firstly defines a road network and route recommendation history. After that, a route recommendation problem is defined. Section 3 proposes a route recommendation method called P-UCT method considering user's individual preferences. Section 4 demonstrates experimental evaluations and Section 5 gives concluding remarks.

2. Route Recommendation Problem Considering User's Individual Preferences

2.1 Road Network

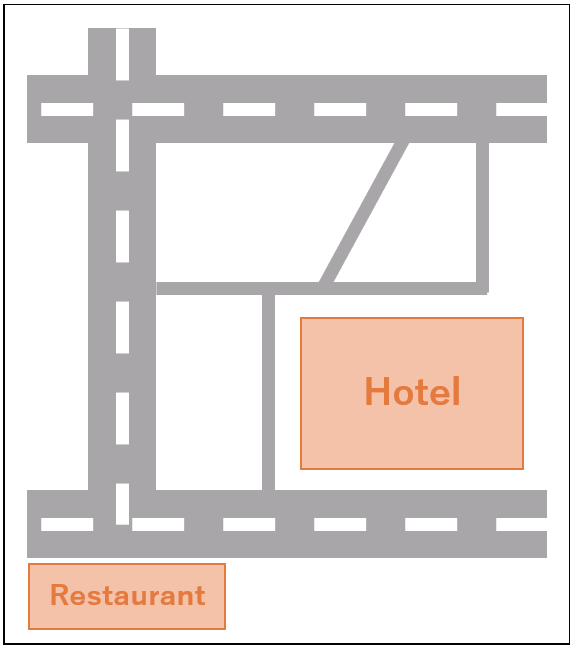
A road network is represented by the graph $G = (V, E)$[*2], where V is a set of nodes showing intersections, corners, and intersections of pedestrian crossings and foot-bridges and E is a set of edges between nodes. A set of landmarks $L = \{l_1, l_2, \cdots, l_k\}$ is given to the road network additionally. Here, $v_s \in V$ is simply called an *intersection node* and $e_u \in E$ is called a *road edge*.

The intersection node $v_s \in V$ has the parameters of latitude *lat*, longitude *lng*, altitude z, a set of visible landmarks $L_{visible} \subseteq L$, where a visible landmark refers to a landmark visible at the target node [16].

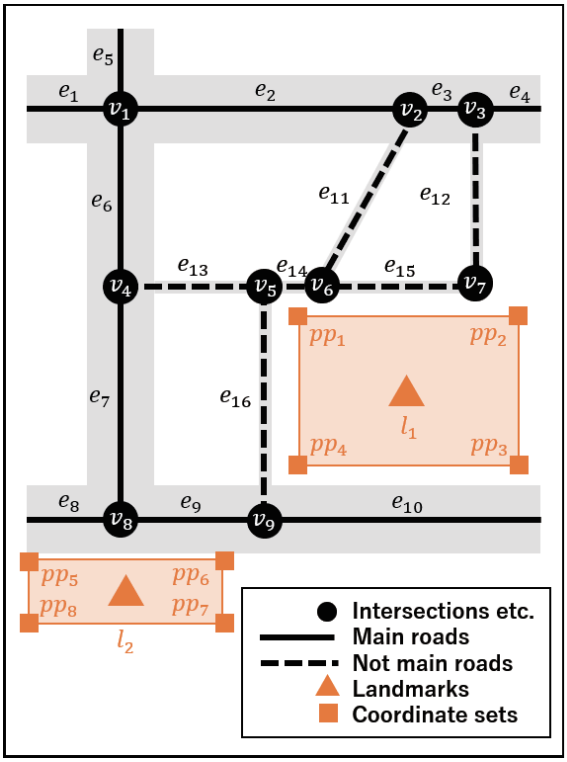
Each edge $e \in E$ is associated with the road width parameter w, which shows e is a main road or not (if $w = 1$, the edge is a main road). Each edge $e \in E$ is also associated with the road type

[*1] The preliminary version of this paper appeared in Ref. [13]. This paper describes the details of the method including the features learned in Support Vector Machine in Section 3.3. Furthermore, we conducted many experiments to confirm the effectivness of the proposed method in Section 4.

[*2] The road network is provided by Zenrin [17].



(a) Map representation.



(b) Road network representation.

Fig. 1 Road network example.

Table 1 Typical landmarks and non-typical landmarks.

Typical landmarks	Government office facilities, Government building, Educational institutions, School, Hospital, Leisure, Traffic, Accommodation building, Commercial building, Target building, General building, Station building
Non-typical landmarks	Other than those above

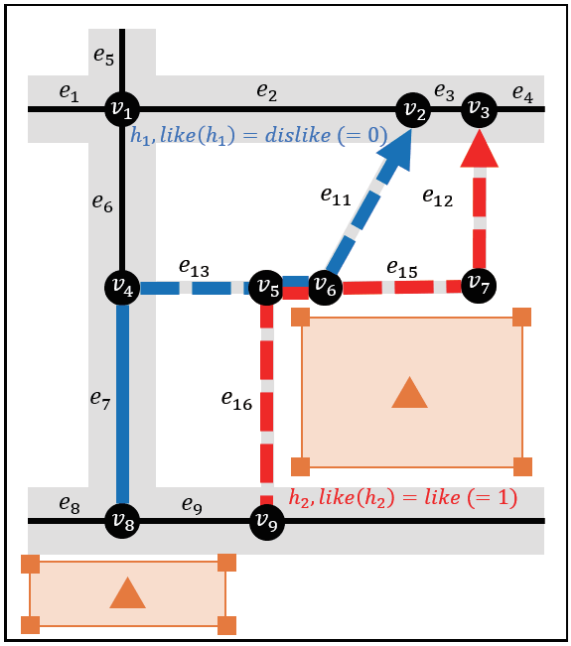


Fig. 2 Route recommendation history example.

ct (ct = either of "sidewalk", "crosswalk", or "none") and the road gradient type sl (sl = either of "stairway", "slope", "step", or "flat").

Each landmark $lt \in L$ has the parameters of the category lc, height h, and a set PP of its coordinates. The landmark category lc indicates whether the landmark is a typical landmark or not (if $lc = 1$, the landmark is a typical landmark). **Table 1** lists the typical landmarks used. PP shows a set of vertex coordinates of the landmark lt, when lt is regarded as a polygon on a map.

Example 1. An example of a road network is shown in **Fig. 1**. Figure 1 (a) shows a map composed of main roads (wide roads) and non-main roads (narrow roads). In Fig. 1 (a), we have two landmarks, a hotel and a restaurant. The road network $G = (V, E)$ showing Fig. 1 (a) is depicted as in Fig. 1 (b), where $V = \{v_1, v_2, \cdots, v_9\}$ and $E = \{e_1, e_2, \cdots, e_{16}\}$. The edges in $\{e_1, \cdots, e_{10}\}$ show the main roads and those in $\{e_{11}, \cdots, e_{16}\}$ show the non-main roads. Whether it is a main road or not is judged by the road width parameter w. l_1 and l_2 show the landmarks of the hotel and the restaurant, respectively. Each landmark has a set of landmark coordinates. l_1 has a set of coordinates $\{pp_1, pp_2, pp_3, pp_4\}$ and l_2 has a set of coordinates $\{pp_5, pp_6, pp_7, pp_8\}$.

2.2 Route Recommendation History

The route recommendation history consists of a set of routes $H = \{h_1, h_2, \cdots, h_u\}$ to which the user u has been recommended beforehand and a parameter $like(h_i)$ $(1 \leq i \leq u)$ representing whether each route h_i is preferable or not to the user u. Every route in the route set H has various start point and end point and the parameter $like(h_i)$ has a two-stage evaluation value: $\{like, dislike\} = \{1, 0\}$.

Figure 2 shows an example of route recommendation history. In Fig. 2, two routes h_1 (blue line) and h_2 (red line) are already recommended to the user u. The route h_1 does not match the preferences of the user u, and the route h_2 matches the preferences of

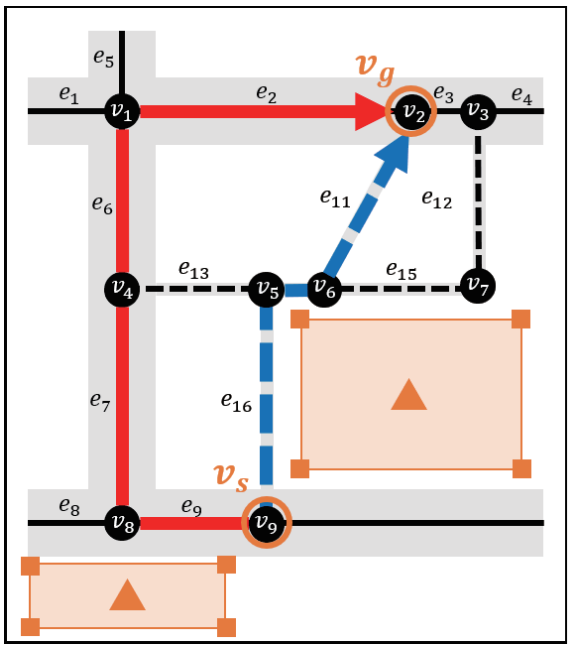


Fig. 3 Route recommendation problem.

the user u.

2.3 Route Recommendation Problem

A route recommendation problem here is defined as follows:

Route recommendation problem: Given a road network $G = (V, E)$, a start point $v_s \in V$ and an end point $v_g \in V$, and a user's route recommendation history, a route recommendation problem is to find a route from v_s to v_g satisfying the user's individual preferences based on his/her route recommendation history.

Example 2. For example, assume that a road network as shown in Fig. 2 is given and a user has the preference that "some amount of route detours are acceptable and it is better to go along a main road" under the route recommendation history. The start point v_s and the end point v_g are also given in **Fig. 3**. In this case, the red route as in Fig. 3 is generated, which is composed of main roads and matches the user's preferences. The blue route in Fig. 3 gives the shortest route from v_s to v_g but it is not preferable to the user.

3. Route Recommendation Method Considering User's Individual Preferences

In the route recommendation problem above, we have to reflect the following two points:

(a) Firstly, we have to take into account individual user's preferences.

(b) Secondly, we have to introduce a mechanism to evaluate an entire route, not a partial route.

In order to realize (a), the proposed method adopts machine learning by a support vector machine (SVM) [18], which is known as a powerful two-class pattern classifier. In the proposed method, a user recognizes every recommended route to be *like* or *dislike* in the route recommendation history and hence the two-class pattern classifier like SVM can be effectively applied to it without explicitly customizing the weights of route elements for every user.

In order to realize (b), the proposed method adopts a UCT (Upper Confidence Tree) approach [10], which is one of typical Monte-Carlo tree search methods. In Monte-Carlo tree search, a large number of Monte-Carlo simulations are performed to obtain a sub-optimal solution where complete routes can be evaluated. We also propose a *P-UCB1 value* to apply Monte-Carlo tree search to route recommendation. The proposed P-UCB1 value can be given to every intersection node in the given road network, which demonstrates how much the intersection node can be preferable for an individual user.

Based on the above discussion, we propose a P-UCT method as follows:

3.1 P-UCT Method

Given a road network $G = (V, E)$, the proposed route recommendation method consists of the following three phases (1)–(3):

Phase (1): Generate a route evaluator

Before performing the route search, extract 20 route features from every route in the route recommendation history and generate a route evaluator using SVM (see Section 3.3 below).

Phase (2): Set up start and end points

Specify the start point $v_s \in V$ and the end point $v_g \in V$ in the road network. Place the pointer v_{now} onto v_s. Initialize the number of Monte-Carlo simulations and the P-UCB1 value for all nodes to 0.

Phase (3): Route search and recommendation

This phase consists of the following five steps (3-1)–(3-5):

Step (3-1): Selection

On the road network, select one node v_{select} with the largest P-UCB1 value among all the nodes adjacent to v_{now}.

Step (3-2): Simulation

Generate a random route from v_{select} to v_g (Monte-Carlo simulation) [*3]. The generated route is called a *playout route*.

Step (3-3): Playout route evaluation

Extract 20 route features from the playout route generated by Step (3-2) and calculate the probability $0 \leq rw_{select} \leq 1$ based on 20 route features using the route evaluator (rw_{select} is called a reward). rw_{select} shows how well the playout route matches the user's preferences.

Step (3-4): Update P-UCB1 value

Update the P-UCB1 value of v_{select} using rw_{select} (see Section 3.2 below). Increase the number of Monte-Carlo simulations for v_{select} by 1.

Step (3-5): Route generation

Repeat Steps (3-1) to (3-4). At the end of Step (3-4), if

[*3] In Step (3-2) of Phase (3), a number of random routes to destination (*playout routes*) are generated by Monte-calro simulation. All the playout routes that reach the destination are evaluated in the next step. However, due to the randomness of route generation, not all routes generated reach the destination in Step (3-2). Since we cannot evaluate a route that does not reach the destination, it is necessary to terminate the simulation in such a case. In order to generate a route without detouring too much, we set the following conditions for the cancellation of the simulation: Let L be the linear distance from v_{select} to v_g. In order to generate a route without detouring too much, we generate a random route from v_{select} to v_g whose distance is not larger than $N_L \times L$, where N_L shows a parameter. Through the preliminary experiments, we tried various N_L values and set up $N_L = 5$ in the experiments in Section 4.

(a) Phase (2): Set up start and end points.
(b) Step (3-1) of Phase (3): Selection.
(c) Step (3-2) of Phase (3): Simulation.
(d) Step (3-3) of Phase (3): Playout route evaluation.
(e) Step (3-4) of Phase (3): Update.
(f) Step (3-5) of Phase (3): Route generation.

Fig. 4 Example of the P-UCT method.

the number of Monte-Carlo simulations for v_{select} reaches the threshold N_{th},[*4] move the pointer v_{now} to v_{select}. If v_{now} reaches v_g, output the route from v_s to v_g along trajectory of the pointer v_{now}.

Example 3. **Figure 4** shows an example of the proposed P-UCT method. We firstly generate a route evaluator based on the route recommendation history (Phase (1), See Section 3.3). After the user specifies the start point and end point, the pointer v_{now} is placed on the start point (Phase (2), Fig. 4 (a)). Since the node v_{now} is adjacent to the two nodes v_1 and v_2, we select the one with a larger P-UCB1 value. In this case, we select v_2, i.e., $v_2 = v_{select}$ (Step (3-1) of Phase (3), Fig. 4 (b)). We randomly generate a route from v_2 to v_g, called a playout route (Step (3-2) of Phase (3), Fig. 4 (c)). In Fig. 4 (c), the dotted blue route shows a playout route. By extracting 20 route features from the playout route, the route evaluator calculates the reward rw_2 ($0 \leq rw_2 \leq 1$), showing how well the playout route matches the user's preferences (Step (3-3) of Phase (3), Fig. 4 (d)). The P-CUB1 value of v_2 is updated by using the rw_2 value and the number of Monte-Carlo simulations in v_2 is increased by 1 (Step (3-4) of Phase (3), Fig. 4 (e)). This process is repeated until the number of Monte-Carlo simulations in the node v_{select} reaches the threshold N. When it reaches N, v_{now} is moved to v_{select} and the route from v_s to v_{select} is determined. For example, if the number of Monte-Calro simulations of v_2 reaches N when v_2 is currently v_{select}, v_{now} is moved to v_2 (Step (3-5) of Phase (3), Fig. 4 (f)).

3.2 P-UCB1 Value

The P-UCB1 value is a node evaluation index that represents the potential of each intersection node $v_j \in V$ based on the UCB1 value [10]. It directly controls the behavior of the P-UCT method proposed in Section 3.1.

[*4] The number of playout routes to apply depends on the threshold N_{th} in Step (3-5) of Phase (3). Through the preliminary experiments, we also tried various N_{th} values and set up $N_{th} = 100$ in the experiments in Section 4.

The initial P-UCB1 value of every intersection node is set to be ∞. Then, based on Ref. [10], the P-UCB1 value of the intersection node v_j is calculated as follows:

$$\text{P-UCB1}(v_j) = \overline{X}_j + \sqrt{\frac{2\log n}{n_j}}. \quad (1)$$

In Eqn. (1), $\overline{X_j}$ is the expected reward value to be returned when the intersection node v_j is selected, n_j shows how many times the intersection node v_j is selected as v_{select}, and n is the sum of the selection numbers (n_j value) of all the intersection nodes in the road network. Note that n and n_j are reset to 0 every time v_{now} is updated.

The expected reward value $\overline{X_j}$ is given by:

$$\overline{X}_j = \frac{\sum_{k=1}^{n_j} rw_j(k)}{n_j}, \quad (2)$$

where $rw_j(k)$ is a reward for v_j for the k-th playout route from v_j.

By using the P-UCB1 value above, we can have the two advantages below:

(1) Intersection nodes with large expected reward values (likely to generate a preferable route) are likely to be selected and Monte-Carlo simulations will be performed for them.

(2) Intersection nodes, not much selected before, are likely to be selected, by adding the second term of Eq. (2).

This is because the P-UCB1 value is composed of (expected value) + (biased value) and the bias value increases when the intersection node is not much selected (i.e., n_j decreases).

3.3 Route Features and a Route Evaluator

3.3.1 Route Features

According to Matsuda's questionnaire survey [3], pedestrian users generally have needs for road types and slope types such as "I want to go through a road with a sidewalk" or "I want to avoid stairs". With regard to the safety characteristics, Matsuda [3] incorporates "presence or absence of sidewalks" and "presence or absence of crosswalks" as safety conditions. Therefore, the proposed method adopts these conditions as feature values and incorporates them as indicators of "Safety" and "Comfort". Yonekura's study [14] reports that it is easy to memorize a route using landmark information. The proposed method adopts these conditions as feature values and incorporates them as indicators of "Difficulty to get lost".

In the proposed method, we do not adopt the feature of the time required directly, but it can be indirectly evaluated by obtaining the feature of the route length ("Time required" later). Accordingly, we consider that the categories that we introduce here correspond to user's preferences.

Based on the these results, we extract 20 road features from a complete route from the start point to the end point as follows:

(A) Time required

1. Route length
   Route length is the total distance from the start point to the end point along the route, which definitely affects the user's preference.

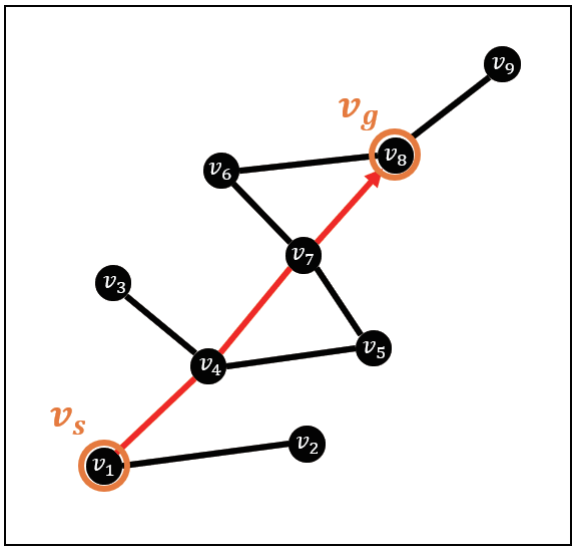


Fig. 5 Route from v_s to v_g.

(B) Difficulty to get lost

2. Number of branches
   The number of branches can be an indicator of the likelihood of being lost in the route, which may much affect the user's preference. The larger the number of intersection nodes in the route becomes, the more we expect that the user is likely to be lost. In addition to that, the larger the number of branches in every intersection node is, the more we expect that the user is likely to be lost.

   Then we define the feature value to be the total number of intersection nodes included in the route and the total number of intersection nodes adjacent to one of the intersection nodes in the route excluding v_g.

   For example, assume that we have a red route from v_s to v_g as shown in **Fig. 5**. In this case, we have four intersection nodes $\{v_1, v_4, v_7, v_8\}$ included in the route. The intersection node adjacent to v_1 (= v_s) is v_2. The intersection nodes adjacent to v_4 are v_3 and v_5. The intersection nodes adjacent to v_7 are v_5 and v_6. The node v_5 is counted twice here. Hence, the number of branches here becomes 9.
3. Number of turns
   According to Yamamoto's study [15], humans recognize directions based on eight directions and the number of turns also definitely affects the user's preference.

   Thus, if the route direction is changed by 22.5 degrees or more, we consider it to be a left/right turn. The number of turns is defined by the total number of left/right turns in the route.
4. Number of visible landmarks
   A visible landmark is a landmark that can be visually seen from each intersection node. In the proposed method, the visible landmarks at each intersection node are determined using the Takeda's method [16].

   The number of visible landmarks is the average number of visible landmarks per intersection node in the route.
5. Number of typical visible landmarks
   Typical visible landmarks are those whose category lc corresponds to typical landmarks of Table 1 in visible land-

marks.

The number of typical visible landmarks is the average number of typical visible landmarks per intersection node in the route.

6. Number of non-typical visible landmarks
Non-typical visible landmarks are those whose category *lc* corresponds to non-typical landmarks of Table 1 in visible landmarks.

The number of non-typical visible landmarks is the average number of non-typical visible landmarks per intersection node in the route.

(C) Safety

7. Route length of sidewalks
Route length of sidewalks is the total distance of the edges whose road type *ct* is "sidewalk" in the route.
8. Number of sidewalk edges
Number of sidewalk edges is the total number of the edges whose road type *ct* is "sidewalk" in the route.
9. Route length of crosswalks
Route length of crosswalks is the total distance of the edges whose road type *ct* is "crosswalk" in the route.
10. Number of crosswalk edges
Number of crosswalks edges is the total number of the edges whose road type *ct* is "crosswalk" in the route.
11. Route length of main roads
Route length of main roads is the total distance of the edges whose road width parameter *w* shows "main road" in the route.
12. Number of highway edges
Number of main road edges is the total number of the edges whose road width parameter *w* is "main road" in the route.

(D) Comfort

13. Route length of stairways
Route length of stairways is the total distance of the edges whose road type *ct* is "stairway" in the route.
14. Number of stairways
Number of stairways is the total number of the edges whose road type *ct* is "stairway" in the route.
15. Route length of slopes
Route length of slopes is the total distance of the edges whose road type *ct* is "slope" in the route.
16. Number of slopes
Number of slopes is the total number of the edges whose road type *ct* is "slope" in the route.
17. Route length of step
Route length of steps is the total distance of the edges whose road type *ct* is "step" in the route.
18. Number of steps
Number of steps is the total number of the edges whose road type *ct* is "step" in the route.
19. Route length of flat roads
Route length of flat roads is the total distance of the edges whose road type *ct* is "flat" in the route.
20. Number of flat road edges
Number of flat road edges is the total number of the edges whose road type *ct* is "flat" in the route.

3.3.2 Route Evaluator

In the proposed method, SVM is used to learn the route features of every recommended route in the route recommendation history and classify every playout route. Here, SVM is used because a user recognizes every recommended route to be *like* (= 1) or *dislike* (= 0) in the route recommendation history and hence the two-class pattern classifier like SVM can be effectively applied to it. The parameters of SVM are optimized by grid search, and are set as the polynomial kernel, the cost parameter $C = 1$, and the hyper parameter $\gamma = 0.1$.

By learning many recommended route data by SVM, we can generate a route evaluator based on SVM. When a test route is given to the learned SVM, it outputs the value how much the route matches the user's preferences ranging from 0.0 to 1.0.

4. Evaluation Experiment

We implemented the proposed method in Python 3 on Intel Core i7 CPU with 16GB RAM and carried out route recommendation experiments. We performed two experiments to confirm the effectiveness of the proposed method.

In Experiment 1, we evaluated the recommended routes for eight males and females aged from 22 to 23 as test users. The maps used were selected around Shinjuku Station and Takadanobaba Station, Tokyo, Japan.

In Experiment 2, we evaluated the recommended routes for six males and females aged from 22 to 55 as test users. In order to investigate whether the route recommendation history collected only in urban areas is also effective for route recommendation in the suburbs, we used the route recommendation history collected only in urban areas and performed route recommendation in both suburbs and urban areas. The maps used were selected Itoshima and Kitakyushu, Fukuoka, Japan as a suburb area. The maps used were selected around Tenjin Station and Kokura Station, Fukuoka, Japan as an urban area.

The example maps used for the experiments are shown in **Fig. 6**. The maps used in the experiments except for Takadanobaba Station area have the area size of 1,000 m × 1,000 m. The map around Takadanobaba Station has the area size of 1,900 m × 2,300 m. The number of intersection nodes, edges, and landmarks of these maps are shown in **Table 2** and **Table 3**.

4.1 Experimental Set Up

Experiment 1 was carried out according to the following procedures (1-1)–(1-5).

(1-1) Fill-out

We prepare random 15 pairs of different start points and end points in the target maps (5 pairs around Shinjuku Sta. and 10 pairs around Takadanobaba Sta.) and a test user writes in his/her preferable routes into the maps. Every pair of start and end points has a linear distance of 300 m to 1,000 m.

(1-2) Setup a route evaluator

Every route that completely matches the written route in Procedure (1-1) is considered to be a positive example for the test user. We also generate random routes and consider them to be negative examples. Then we learn these positive and negative examples using the SVM classifier and set up a

Table 2 Map data information of Experiment 1.

Map data	# of intersection nodes	# of edges	# of landmarks
Shinjuku Station area	4881	5611	66
Takadanobaba Station area	10614	11936	130

Table 3 Map data information of Experiment 2.

Map data		# of intersection nodes	# of edges	# of landmarks
Suburb	Itoshima area	809	832	0
	Kitakyusyu area	522	534	0
Urban	Tenjin Station area	4938	5786	105
	Kokura Station area	3154	3621	51

(a) Shinjuku station area.

(b) Takadanobaba station area.

Fig. 6 The example maps used for the experiments.

route evaluator.

In this experiment, the number of routes learned by SVM is 10, 20, and 30, in which the number of positive examples and the number of negative examples [*5] are always the same (for example, when we learn 10 routes by SVM, 5 positive examples and 5 negative examples are learned) [*6].

(1-3) Route recommendation

After having learned the fixed number of positive and negative examples by SVM in Procedure (1-2), a new pair of start and end points which linear distance is 300 m–1,000 m, not used in learning, are specified in the target map and a route between them is recommended to the test user by the proposed P-UCT method.

(1-4) Evaluation

The test user evaluates how well the route recommended by the proposed method matches the test user's preferences. The evaluation is done by the five-scale rating. The lowest evaluation score is 0 and the highest evaluation score is 4. Note that a user cannot well evaluate the recommended route to be "like" or "dislike", if detailed factors such as landmark and road conditions as discussed in Section 3.3.1 are not provided. However, in Experiment 1, the test users commute to or live in the target map areas and the test users well know every route on the map areas. Hence they can evaluate every route by assuming the actual road conditions with only map information.

(1-5) Comparison with the existing method

For the comparison purpose, we also carry out route recommendation using the method proposed in Ref. [3] [*7]. In the same way, the test user evaluates the generated route by Ref. [3] using the five-scale rating.

In order to reflect user's preferences, one of the easiest ways is that we perform a large-scale questionnaire survey for each user from the viewpoints of the route features such as in Section 3.3.1 and construct a route recommendation system customized for every user. But it must be impractical because performing such a large-scale questionnaire survey requires much time and effort and it must be a burden for

[*5] The route data learned as a negative example is a randomly generated route from the specified start point to the end point. The method of generating this random route is simple: select one node randomly from the intersection nodes connected to the start point to proceed, and then repeat the process of selecting one randomly from the intersection nodes connected to the current intersection node to proceed in the same way. We simply consider the routes generated above to be negative examples in Experiment 1 and Experiment 2.

[*6] If we evaluate routes using SVM on data with unbalanced number of positive and negative examples, the results are generally pulled towards the class with more ones (to the positive direction if there are more positive ones and to the negative direction if there are more negative ones). Therefore, it can be true that the balanced examples are used in route learning. As in the footnote *5, we can easily generate negative examples and hence we prepare the same number of positive and negative examples in the experiments.

[*7] Ref. [3] is based on Dijkstra method, but when each path element (e.g., slope, crosswalk) is passed, the actual distance is multiplied by the pathfinding parameter associated with each path element. The pathfinding parameters are predetermined by the questionnaire. For example, the slope parameter is 1.418, which results in a larger distance than the actual slope distance.

Table 4 Experimental results in Experiment 1.

Route recommendation method		Score by test user								Average score
		A	B	C	D	E	F	G	H	
Proposed method	Number of learning routes = 10	0	0	0	3	0	2	2	0	0.875
	Number of learning routes = 20	0	2	4	3	2	1	4	3	2.375
	Number of learning routes = 30	4	4	4	4	0	1	4	3	3.000
Existing method [3]		2	4	4	0	0	2	1	2	1.875

every user. Hence, in Ref. [3], the general weights are designed to route elements based on the large-scale surveys for many users at once and the best preferable routes are generated to almost all the users without explicitly customizing the weights of route elements for every user. How to weigh each factor is another concern and the method [3] solves this problem effectively by using the Dijkstra's method. This is quite a practical approach.

On the other hand, the proposed method does not explicitly obtain a large-scale questionnaire survey for route recommendation for every user but instead just asks whether the route is like or dislike in the route history. The proposed method learns the route based on the 20 route features by SVM but every user does not have to characterize each route feature in the questionnaire.

Both of the approaches are quite similar from the viewpoint of reducing the users' burden but provides the route preferable to users. In this sense, we believe that the comparison is fair.

Experiment 2 was carried out according to the following procedures (2-1)–(2-3).

(2-1) Setup a route evaluator

We obtain 50 positive and negative examples (route recommendation histories) in only urban areas per test user by the same procedure as in Experiment 1. After that, we learn these positive and negative examples using the SVM classifier and set up a route evaluator.

(2-2) Route recommendation

After having learned 50 positive and negative examples by SVM in Procedure (2-1), a new pair of start and end points which linear distance is 1,000 m, not used in learning, is specified in the target map and a route between them is recommended to the test user by the proposed P-UCT method. At this time, we perform route recommendation in both suburb and urban areas 40 times each.

(2-3) Evaluation

The test user evaluates how well the route recommended by the proposed method matches the test user's preferences. The evaluation is done by the five-scale rating. The lowest evaluation score is 0 and the highest evaluation score is 4. In Experiment 2, the test users also commute to or live in the target map areas and the test users well know every route on the map areas. Hence they can evaluate every route by assuming the actual road conditions with only map information.

4.2 Experimental Results

4.2.1 Result of Experiment 1

The results in Experiment 1 are shown in **Table 4**. The number of learning routes is the routes learned in Procedure (2) of Section 4.1 (the sum of positive and negative examples), and A–H show the test users. The numbers below the test users show the five-scale scores which each test user gives to every recommended route. As shown in Table 4, the average score of the proposed method is 0.875, 2.375, and 3.000 when 10, 20, and 30 learning routes are given, respectively. On the other hand, the average score of the existing method [3] is 1.875.

As an example, the route recommendation results of the test user A are shown in **Figs. 7–10**. When the number of learning routes is 10 or 20, the evaluation score becomes 0 since the proposed method recommends complicated and hard-to-understand routes to the test user. When the number of learning routes reaches 30, the evaluation score becomes 4 since the proposed method recommends the route which is almost the same as the one that the test user usually uses. The route recommended by the existing method [3] (Fig. 10) has a small number of turns but is not easy to understand. The test user gives the score of 2 to this route.

Table 6 summarizes the feature values in Section 3.3.1 for Fig. 9, Fig. 10 and the route that the test user A usually uses. Each value in Table 6 shows the average value for every category. As Table 6 indicates, the proposed method recommends the route very close to the one that the test user A usually uses quantitatively, comparing to the existing method.

The purpose of Experiment 1 is to investigate how many routes in the recommendation history we need to learn to obtain reasonably good results. We can clearly obtain a high-scored result when the number of routes in the route history is increased. Particularly, when we learn 30 routes in the proposed method, the scores of the obtained routes become the highest in almost all the cases and higher than those of the existing method in almost all the cases. As in Figs. 7–10, the obtained routes must be good enough when we learn 30 routes. We cannot say that 20–30 routes are sufficient theoretically in machine learning but these results of Table 3 and Figs. 7–10 show that the proposed method is superior to the existing method when learning 20 or more routes at least.

However, it must be important to see if the number of routes is increased more. **Table 7** summarizes the results on the additional experiment. In this experiment, the set up is the same as Experiment 1 but the number of routes learned is increased to up to 100. In this case, the test user H participates in this experiment, since the score of the test user H does not reach the highest level when 30 routes are learned. Even if the number of the routes learned is increased, the score is not changed too much. We consider that learning 30–50 routes can lead reasonably good results.

Overall, when the number of learning routes is 20 or 30, the results show that the proposed method outperforms the existing method in most cases. The results indicate that, if the test user

Table 5 Experimental results in Experiment 2.

Map used for route recommendation	Average score by test user						Overall average
	I	J	K	L	M	N	
Suburbs	3.20	3.30	2.95	3.15	2.95	3.15	3.117
Urban areas	3.35	3.20	2.85	3.40	3.15	3.10	3.175

Table 6 Comparison of features for each route.

Feature categories	Time required	Difficulty to get lost	Safety	Comfort
Route by proposed method (**Fig. 9**)	0.87	1.03	0.20	0.02
Route by existing method (**Fig. 10**)	0.76	1.15	0.20	0.02
Route usually taken	0.86	1.05	0.19	0.02

Fig. 7 The route recommendation result to the test user A by the proposed method (10 learning routes).

Fig. 8 The route recommendation result to the test user A by the proposed method (20 learning routes).

prefers main roads and a small number of left/right turns, the proposed method can easily find out those preferable routes, even if the number of learned routes is small. However, if the test user prefers a route which goes through a complicated residential area or park, the proposed method cannot always generate a satisfactory route. This is because the proposed method is based on Monte-Carlo simulations and cannot always obtain the best results when too complicated routes are preferable.

Fig. 9 The route recommendation result to the test user A by the proposed method (30 learning routes).

Fig. 10 The route recommendation result to the test user A by the existing method [3].

4.2.2 Result of Experiment 2

The results in Experiment 2 are shown in **Table 5**. I–N show the test users. As shown in Table 5, the average of the evaluation scores of the route recommendation results in urban areas was higher, but the difference was just 0.058 points.

As an example, the route recommendation results of the test user I are shown in **Figs. 11** and **12**. Figure 11 shows the route recommendation result in a suburb area and Fig. 12 shows one in an urban area. In both cases, the evaluation scores become

Table 7 The test user H's rating of the recommended routes.

Method	Proposed method (# of learning routes)						Existing method [3]
	10	20	30	50	70	100	
Score	0	3	3	3	3	3	3

Fig. 11 The route recommendation result in a suburb area.

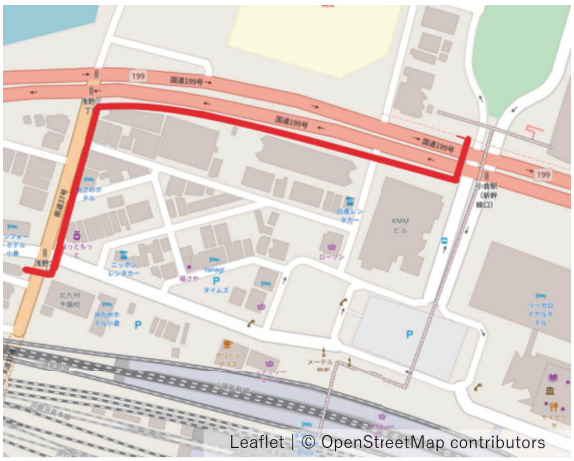


Fig. 12 The route recommendation result in an urban area.

4 since the proposed method recommends easy-to-understand routes with few turns.

With regard to the differences in road characteristics between suburb and urban roads, the main differences are in the number of intersection nodes and landmarks, as shown in Table 3. Roughly saying, the number of intersection nodes in the suburb areas is 1/4–1/6 compared to the urban areas. The number of edges in the suburb areas is 1/10–1/4 compared to the urban areas. There may be no landmarks in the suburb areas. Based on this premise, Experiment 2 is conducted with the proposed method learning only urban routes. The results of Experiment 2 show that the recommendation accuracy is almost the same in both suburb and urban areas, as shown in Table 5. As in Table 5, the score of the suburb areas is higher in some users but the score of the urban areas is higher in other users. The difference between them is very small. "0.058" shows the difference between the average scores in suburb and urban areas in all the users, which we believe is small enough. Table 5 indicates that the proposed method is effective even in areas with different characteristics.

This result shows that: even when the recommended route history collected only in urban areas was used for route recommendation in the suburb area, the recommended route reflected the user's individual preferences.

Table 8 Average computation time of the route recommendation in the proposed method.

Linear distance between start and end points	Average computation time
300 m	125 s
500 m	385 s
700 m	1,158 s
1,000 m	3,027 s

4.3 Computation Time

In order to justify the effectiveness of the proposed method, the computation time to obtain every converged result of the proposed method is summarized in **Table 8**. The computation time is the average of 20 route recommendations for every linear distance, varying start and end points.

As in Table 8, the proposed method requires 100–3,000 seconds to obtain a recommended route currently. The computation time is approximately 120 seconds when the linear distance between start and end points is 300 m, which may be acceptable, but as the linear distance becomes larger, the computation time also becomes larger. Since the implementation of the proposed method is just the first trial to confirm the effectiveness of the proposed method, the computation time can be reduced by implementing the method more carefully. How to reduce the computation time so that the quality of recommended routes is not degraded is one of the most important future works.

5. Conclusion

In this paper, we proposed a route recommendation method, called *P-UCT method*, considering individual user's preferences utilizing Monte-Carlo tree search. The experimental results demonstrate that the average score of the proposed method becomes 3.00 (the full score is 4.00) when 30 routes are learned beforehand, while that of the existing method becomes 1.875. We also demonstrate that the proposed method provides the recommended route reflecting the user's individual preferences even if it learns the recommended route history of areas in different situations.

In the future, we will reduce the computation time of the proposed method so that the quality of recommended routes is not degraded.

References

[1] Yahoo Japan Corporation: Yahoo! MAP, available from ⟨https://map.yahoo.co.jp/⟩ (accessed 2020-03-25).
[2] NAVITIME JAPAN, available from ⟨https://www.navitime.co.jp/⟩ (accessed 2020-03-25).
[3] Matsuda, M., Sugiyama, H. and Doi, M.: A personalized route guidance system for pedestrians (in japanese), *Trans. IEICE*, Vol.87, No.9, pp.132–139 (2004).
[4] Hara, K. and Kanoh, H.: Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network, *Proc. 10th Annual Conference on Genetic and Evolutionary Computation*, pp.657–664 (2008).
[5] Kurata, Y. and Shinagawa, Y.: CT-Planner5: A computer-aided tour planning service which profits both tourists and destinations, *Proc. Workshop on Tourism Recommender Systems*, Vol.15, pp.35–42 (2015).
[6] Sawayanagi, Y. and Hamakawa, R.: Calculation of the user optimum path in mobile navigation with ambiguous destinations, *IPSJ Technical Report, MPS-73*, pp.25–28 (2009).

[7] Kanoh, H., Nakamura, N. and Nakamura, T.: Route selection with unspecified sites using knowledge based genetic algorithm, *Trans. Japanese Society for Artificial Intelligence*, Vol.17, pp.145–152 (2002).

[8] Akasaka, Y. and Onisawa, T.: Individualized pedestrian navigation using fuzzy measures and integrals, *Proc. 2005 IEEE International Conference on Systems, Man and Cybernetics*, Vol.2, pp.1461–1466 (2005).

[9] Teng, L., Izumi, T., Lu, X. and Wakui, F.: A Method of the Optimum Route Search by Fuzzy-AHP, *IEEJ Trans. Electronics, Information and Systems*, Vol.133, pp.1269–1276 (2013).

[10] Jouini, W., Ernst, D., Moy, C. and Palicot, J.: Upper Confidence Bound Based Decision Making Strategies and Dynamic Spectrum Access, *2010 IEEE International Conference on Communications*, pp.1–5 (2010).

[11] Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver D., Szepesvári, C. and Teytaud, O.: The grand challenge of computer Go: Monte Carlo tree search and extensions, *Comm. ACM*, Vol.55, No.3, pp.106–113 (2012).

[12] Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search, *Proc. International Conference on Computers and Games*, pp.72–83 (2006).

[13] Ishizaki, Y., Takayama, T. and Togawa, N.: A route recommendation method based on personal preferences by Monte-Carlo tree search, *Proc. IEEE International Conference on Consumer Electronics in Berlin* (*ICCE-Berlin*), pp.404–409 (2019).

[14] Yonekura, R., Morinaga, H., Wakamiya, S., Akagi, Y., Ono, S., Kawai, Y. and Kawasaki, H.: Navigation system without explicit route presentation using point, line and area based landmarks (in japanese), *Proc. IPSJ Interaction* (2015).

[15] Yamamoto, N. and Okabe, A.: Qualitative spatial reasoning about directions in a corridor containing a single turn, *MERA Journal*, Vol.7, No.2, pp.11–20 (2002).

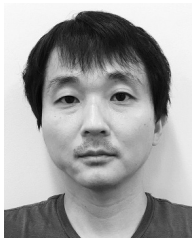
[16] Takeda, K., Nitta, T., Shindou, D., Yanagisawa, M. and Togawa, N.: A visible corner-landmark based route finding algorithm for pedestrian navigation, *Proc. 2015 IEEE 4th Global Conference on Consumer Electronics* (*GCCE*), pp.601–602 (2015).

[17] Zenrin, available from ⟨https://www.zenrin.co.jp/english/index.html⟩ (accessed 2020-03-25).

[18] Adankon M.M. and Cheriet M.: *Support vector machine*, pp.1303–1308, Springer (2009).

Yuta Ishizaki received his B. Eng. degree from Waseda University in 2019 in computer science and engineering, where he is working towards M. Eng. degree. His research interests include intelligent transportation system.

Yurie Koyama received her B. Eng. degree from Waseda University in 2020 in computer science and engineering. Her research interests include intelligent transportation system.



Toshinori Takayama received his bachelor degree from Yokohama City University in 1998 in Philosophy. He is presently working in the ZENRIN DataCom Co., Ltd.

Nozomu Togawa received his B. Eng., M. Eng., and Dr. Eng. degrees from Waseda University in 1992, 1994, and 1997, respectively, all in electrical engineering. He is presently a Professor in the Department of Computer Science and Communications Engineering, Waseda University. His research interests are intelligent transportation system, ingtrated system design, graph theory, and computational geometry. He is a member of IEEE and IEICE.