# Road Segment Interpolation for Incomplete Road Data

Yuya Sasaki
*Graduate School of Information Science and Technology*
Osaka University
Osaka, Japan
sasaki@ist.osaka-u.ac.jp

Jiahao Yu
*Graduate School of Information Science*
Nagoya University
Nagoya, Japan
yu2014.db.is@gmail.com

Yoshiharu Ishikawa
*Graduate School of Informatics*
Nagoya University
Nagoya, Japan
ishikawa@i.nagoya-u.ac.jp

***Abstract*—Road data is fundamental information for location-based services. We trust that the road data is complete to represent an actual road network when we develop the location-based services. However, road data may be incomplete due to update delays, and thus location-based services may not provide useful results. Several algorithms have been proposed to automatically update road data. In this paper, we study interpolation of missing road segments by using vehicle trajectory data. We can find missing road segments from the trajectories because vehicles may pass through road segments that are not included in road data. However, trajectories are inherently noisy due to GPS errors. Hence, we cannot easily interpolate appropriate road segments. We propose an algorithm based on map matching and clustering techniques for achieving accurate and comprehensive interpolation. Our algorithm first detects trajectories that are probably on missing road segments. It then clusters the trajectories by DBSCAN and integrates the trajectories for interpolating the road data. Through the experiments using real incomplete road data and trajectory data, we verify that our algorithm effectively interpolates the missing road segments.**

***Index Terms*—Road networks, Map matching, DBSCAN**

## I. Introduction

Recent devices such as smart phones and car navigation systems are commonly equipped with GPS systems. We can easily acquire our current locations. This development has triggered the advance of location-based services such as route search and location recommendation [14]. Location-based services often use road data to search for routes and/or calculate the distances between our current locations and the destinations. The road data is essential for wide location-based services.

For developing location-based services, we trust that the road data is complete to represent the actual road network. However, road data may be incomplete due to update delays. For example, road data provided by OpenStreetMap[1] is manually updated by voluntary users, and thus it may have update delays and omissions. Another example is road data in developing countries which are rarely updated due to human resource shortage. Figure 1 shows road data in Beijing extracted from OpenStreetMap and trajectories of taxis provided by the T-drive project [15]. Black points and lines denote the road data, and red points denote GPS points of the trajectories. A trajectory is through the area where there is no road segments in the road data. Road segments probably exist close to the trajectory in the actual road network.

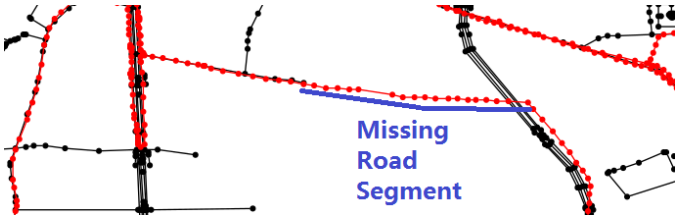



Fig. 1: Example of incomplete road network and trajectory data.

Incomplete road data becomes a problem for both users and businesses. If we use incomplete road data, location-based services may output unexpected and incorrect results which cause loss of the quality of service. Thus, many works attempt to solve the problem by automatically updating the road data based on vehicle trajectories [3], [8]. We have three incomplete patterns; unnecessary, inaccurate, and missing.

- Unnecessary: road data has unnecessary road segments and/or intersections that are not existing in the actual road network.
- Inaccurate: locations of road segments and/or intersections are inaccurate.
- Missing: road data misses some road segments and/or intersections.

Approaches for updating such incomplete patterns are different from each other. Unnecessary road segments are easily deleted from the road data when vehicles are not through the road segments for a long time. Inaccurate road segments is updated when trajectories and road segments are statistically different. Updating inaccurate road segments is the main line of incomplete road data, and thus several algorithms have been proposed such as fixing centerlines [13]. To the best of our knowledge, there are no literature for interpolating missing road segments to the road data. In addition, approaches for updating other patterns do not update missing road segments.

Therefore, in this paper, we propose an algorithm to interpolate missing road segments on incomplete road data by using

[1]https://www.openstreetmap.org

978-1-5386-7789-6/19/$31.00 ©2019 IEEE

vehicles trajectories. Since vehicles pass through missing road segments like Figure 1, we can find missing road segments from the trajectories. However, trajectories are inherently noisy due to GPS errors, and thus it is not appropriate to use trajectories themselves as missing road segments. This causes three challenges to appropriately interpolate road segments. First, it is difficult to accurately detect trajectories that may be on missing road segments. We need to detect trajectories that pass on missing road segments with high probabilities. Second, if we interpolate trajectories as road segments, the road segments probably take unrealistic shapes. Hence, we need to modify shapes of trajectories to be realistic. Third, if many trajectories are detected and they are close to each other, we may unnecessarily interpolate many road segments while they pass the same road segments. For avoiding unnecessary interpolation, we need to integrate the trajectories.

For tackling these challenges, our algorithm consists of four steps: (1) extracting trajectories as candidates of missing road segments by a map matching algorithm [10], (2) simplifying the candidates of trajectories to take realistic shapes, (3) clustering the candidates by DBSCAN [7], and (4) integrating the candidates close to each other and interpolating them to the road data. Through the experiments with real incomplete road data OpenStreetMap and trajectory data T-drive data, we verify that our algorithm effectively interpolates road segments. Our algorithm interpolates several shapes of road segments from straight to complex shapes.

The main contributions of this paper are as follows.

- This is the first work for interpolating missing road segments of incomplete road data.
- We propose an algorithm for interpolating road segments.
- We verify our algorithm works well by the experiments with the real incomplete data and trajectory data.

The remainder of this paper is organized as follows. Section II introduces the related work. Section III describes the preliminaries. Section IV presents our algorithm. Section V presents the results obtained from the experiments, and Section VI summarizes the paper.

## II. Related work

Interpolation of road segments from GPS trajectories is related to two lines of research; (1) constructing road data and (2) updating road data.

First, we review some works for constructing (or inferencing) road data from GPS trajectories. This research topic focuses on construction of comprehensive and accurate road data. Edelkamp and Schrödl [6] apply k-means algorithm to cluster GPS points. Then, in order to construct road data, it connects clusters that GPS trajectories pass through. Davies et al. [5] propose a 2D histogram-based algorithm, which divides an area into grid cells and counts the number of GPS points in each cell. When cells have larger numbers of GPS points than the threshold, they decide that the cells include road segments or intersections. Biagioni and Eriksson [1] extend the algorithm [5]. It gradually decreases the threshold and avoids outputting spurious road segments. Cao and Krumm [2] propose an algorithm for repelling and attracting GPS trajectories to construct several lanes of road segments. GPS trajectories are repelled if the trajectories move the opposite direction each other. On the other hand, if the trajectories move in the same direction, they are attracted. Fathi and Krumm [8] propose a junction detector that is trained by GPS points to learn patterns of GPS points on intersections. Moreover, Chen et al. [3] use additional information for learning such as types of intersections to infer the correct map representation. Chen et al. [4] also propose a junction detector from GPS trajectories, but it does not require any training. This algorithm extends DBSCAN with moving directions for detecting intersections. This line of research needs a large amount of GPS trajectories to construct road data. If the number of GPS points is small in an area, they ignore the road segments and intersections in the area for constructing accurate road data. Missing road segments in road data are typically minor ones, and their traffic volumes are relatively small. As a result, these works cannot interpolate missing road segments.

Updating (or refining) road data focuses on updating an existing road data using GPS trajectories. Rogers et al. [13] and Guo et al. [9] propose frameworks to refine the centerlines of the road segments. These frameworks infer the centerlines of the road segments from GPS trajectories. The former framework [13] refines the centerlines by the average of the existing road data and the closest trajectory point, weighted by the confidence in the road data and the trajectory. The latter framework [9] calculates the centerlines based on a distribution model of GPS points. Although these works update the road data, they do not assume missing road segments.

TABLE I: Summary of notations

| Symbol | Meaning |
|---|---|
| $p$ | GPS point |
| $t$ | Trajectory (sequence of GPS points) |
| $\mathbb{T}$ | Set of trajectory |
| $R = (V_R, E_R)$ | Incomplete road data |
| $V_R$ | Set of intersections and curve points |
| $E_R$ | Set of road segments |
| $dist_R(v, v')$ | The shortest road network distance between $v$ and $v'$ |
| $dist_E(l, l')$ | The Euclidean distance between $l$ and $l'$ |

## III. Preliminaries

In this section, we explain our notations and two techniques; map matching and DBSCAN, as preliminaries of our algorithm.

### A. Trajectories

We here define several terms required to introduce our problem. Table I summarizes the notations used in this paper.

*Definition 1:* (**GPS point**) A GPS point $p$ has location $l$, time stamp, and an identifier of device. The location $l$ is denoted by latitude and longitude. Since GPS points include

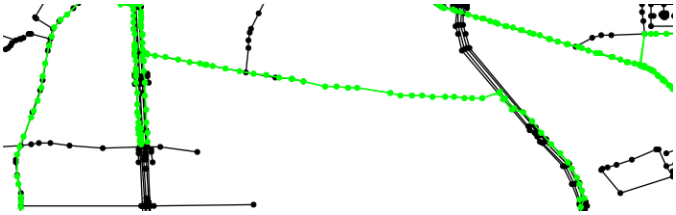


Fig. 2: Result of map matching for incomplete road network.

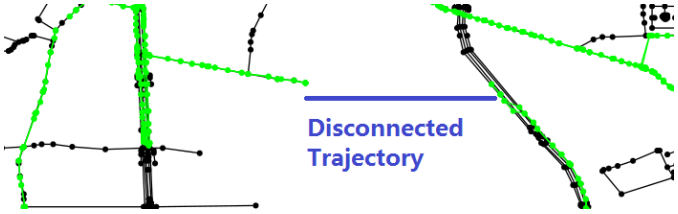



Fig. 3: Result of map matching not for incomplete road network.

noise, $l$ may have some errors. We do not know the ranges of errors. □

*Definition 2:* (**Trajectory**) A trajectory $t = \{p_1, p_2 \cdots, p_n\}$ is a sequence of GPS points, where the number of GPS points in $t$ is $n$. We denote the number of GPS points in $t$ by $|t|$. The GPS points are in ascending order of the time stamp, and all GPS points have the same device identifier. □

We denote the set of trajectories by $\mathbb{T}$.

*Definition 3:* (**Road data**) Road data is represented by an undirected graph $R = (V_R, E_R)$, where $V_R$ and $E_R \in V_R \times V_R$ represent the set of intersections and curve points, and the set of road segments, respectively. Each vertex $v \in V_R$ has accurate location. The road data is incomplete, and thus it may miss some intersections, curve points, and road segments. □

We denote the location of $p$ and $v$ by $p.l$ and $v.l$, respectively. From the locations of intersections and curve points, we can calculate the distance between a GPS point and a road segment. We define two distance: $dist_R(u, v)$ and $dist_E(l, l')$. $dist_R(v, v')$ denotes the shortest road network distance between $v$ and $v'$. On the other hand, $dist_E(l, l')$ denotes the Euclidean distance between $l$ and $l'$. We compute the length $d(t)$ of trajectory $t$ by the following equation:

$$d(t) = \sum_{i=1}^{|t|-1} dist_E(p_i.l, p_{i+1}.l). \quad (1)$$

In this paper, we define the problem as follows.

PROBLEM DEFINITION *Given a set of trajectories $\mathbb{T}$ and a road data $R$, we interpolate missing road segments into $R$ by using $\mathbb{T}$.*

### B. Map matching

Map matching maps a set of GPS points with errors to the corresponding locations on the road segments. However, if the road data is incomplete, general map matching does not work well. Figures 2 and 3 show the result of map matching in Figure 1. Figure 2 shows the result of a map matching technique for incomplete road data [10] while Figure 3 shows the result of a different map matching technique for complete road data [11]. Comparing two results, the map matching technique for incomplete road data outputs more natural result than it for complete road data.

We use the map matching technique for incomplete road data [10]. We briefly show the algorithm. The map matching algorithm is based on Hidden Markov model-based algorithm

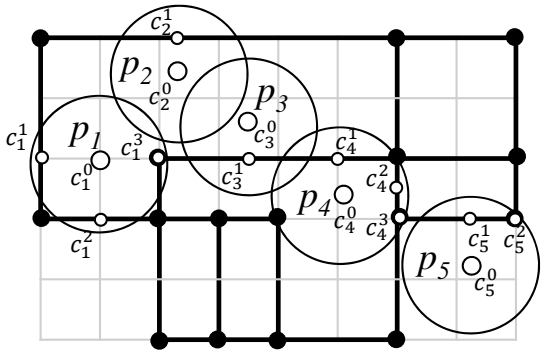



Fig. 4: Candidate selection of map matching

[12]. It first selects a set of candidates $C_i = \{c_i^0, c_i^1, \cdots, c_i^K\}$ for each $p_i \in t$, where $K$ is a parameter for setting the maximum number of candidates. Candidates $c_i^1, \cdots c_i^K$ have locations on the $K$ closest road segments within a certain range. $c_i^0$ is a special point that has a location of $p_i$ itself (the same location of the GPS point). Then, this algorithm calculates the shortest road network distance and the Euclidean distance between all candidates of $p_i$ and $p_{i+1}$. That is, we calculate $dist_R(c_i^j, c_{i+1}^j)$ and $dist_E(c_i^j.l, c_{i+1}^j.l)$ for $0 \leq i \leq n-1$ and $0 \leq j \leq K$. We construct a complete $n$-partite graph whose vertices and edges represent the set of candidates of GPS points and transitions between vertices. Weights of edges represent *transition probabilities* that are computed from their distances between vertices so that closer distances have higher values. Finally, the path on the $n$-partite graph with the largest transition probability is a matching result from $p_1$ to $p_n$. In this paper, we call sub-trajectories that are through GPS points *off-road* trajectories.

Figure 4 shows road data and five GPS points. The GPS point $p_3$ may be through a missing road segment. When we set $K$ as three, each candidate has at most four candidates. Then we construct five-partite graph (see Figure 5). The path $(c_1^1, c_2^1, c_3^0, p_4^1, p_5^1)$ with the largest probability becomes the matching result. The off-road trajectory is $(c_2^1, c_3^0, p_4^1)$.

### C. DBSCAN

DBSCAN [7] is a popular clustering technique and clusters points if they are densely connected. DBSCAN has two parameters; $minPts$ and $\epsilon$. If a points has at least $minPts$ points (including itself) within a circle whose radio is $\epsilon$, the point is called a *core point*. Each point closer to a core point than $\epsilon$ belongs the same cluster. Moreover, if two core points

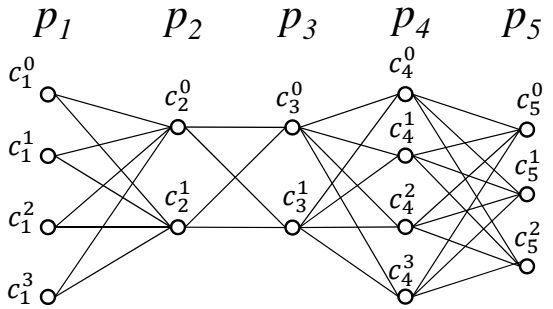



Fig. 5: $n$-partite graph of map matching

are located within $\epsilon$, the core points belong the same cluster. As a result, DBSCAN clusters densely connected points.

## IV. Proposal

We present our algorithm for interpolating missing road segments. We first describe its design policy and then explain our algorithm in detail.

### A. Design Policy

We have two approaches for interpolating missing road segments; (1) the *construction approach* and (2) the *detection approach*. The construction approach first constructs a road data from trajectories without using any road data, and then integrate constructed and existing road data. The detection approach detects candidates of missing road segments using trajectories and existing road data, and then interpolate the candidates into the existing road data. For interpolating missing road segments, we need to consider characteristics of missing road segments. Missing road segments are typically minor roads because major roads are preferentially updated. The volume of traffic on minor roads are small compared to major roads. Therefore, we have to detect the missing road segments from a small number of trajectories. The construction of road data [8] basically assumes large amount of trajectories to construct comprehensive and accurate road data. In this case, missing road segments may not appear in the constructed road data because they are not reliable due to a small number of trajectories and GPS errors. On the other hand, the detection approach can find the minor roads from a small number of trajectories. Thus, we select the detection approach as the basis of our algorithm.

Our algorithm effectively interpolates missing road segments by using trajectories. For accurately interpolating missing road segments, we have three problems; (1) how to detect candidates of missing road segments, (2) how to modify the shapes of trajectories for natural interpolation, and (3) how to integrate road segments that are close to each other. Our algorithm solves these problems. For the first problem, we detect candidates of missing road segments by using a map matching technique for incomplete road data. For the second problem, we modify the trajectories to take realistic shapes of road segments by the proposed simplifying technique. For the third problem, we cluster the candidates, and then integrate the candidates in the same cluster.

### B. Overview

Figure 6 shows an overview of our algorithm. Our algorithm consists of four steps: (1) Extraction, (2) Simplification, (3) Clustering, and (4) Integration. The extraction step extracts candidates of trajectories that may be on missing road segments by the map matching algorithm [10]. The simplification step simplifies the extracted trajectories to realistic shapes. The clustering step clusters the simplified trajectories close to each other. Finally, the integration step integrates the simplified trajectories within the same cluster, and then interpolates it into the road data.

### C. Extraction

We extract sub-trajectories that may be on the missing road segments. For this purpose, we use a map matching algorithm for incomplete road data [10]. The algorithm basically maps each GPS point to a point on a road segment, while it outputs each GPS point itself if the mapped points on road segments are quite distant on the road network distance. We extract off-road trajectories which can be considered as trajectories in missing road segments. However, off-road trajectories may be caused due to GPS errors. Since short off-road trajectories with small numbers of points are caused by GPS errors with high probabilities, we use off-road trajectories with the numbers of points larger than $\tau$ and whose lengths are larger than $\psi$ as the candidates of missing road segments. We extract reliable candidates of missing road segments by the extraction step.

### D. Simplification

It is not practical to directly interpolate off-road trajectories into road data because off-road trajectories usually do not have realistic shapes. More specifically, off-road trajectories are often zigzag due to GPS errors. Our algorithm simplifies off-road trajectories to change their shapes to realistic ones. Since zigzag road segments are very rare in actual road networks, we approximate off-road trajectories by the least number of points to take straight lines. Our simplification technique repeatedly judges whether trajectories are straight or not. It first calculates angles and distances to a line between $p_1$ and $p_n$ for all points in the off-road trajectory. If all the angles and the distances are smaller than $\theta$ and $\delta$, respectively, we judge that the off-road trajectory is straight. If the off-road trajectory is not straight, we remove the last point from the trajectory. We iteratively compute the off-road trajectory until it becomes straight, and then we simplify the removed points in the same way. By connecting all the straight lines, we obtain simplified off-road trajectories.

Figure 7 shows an example of straight line judgment. In the figure, a trajectory has six GPS points, where 1 and 6 are the first and last GPS points in the trajectory, respectively. Each GPS points compare the distance and the angle to the line between 1 and 6 (red line). If the angles and distances are smaller than $\theta$ and $\delta$, respectively, we judge that the trajectory is straight.

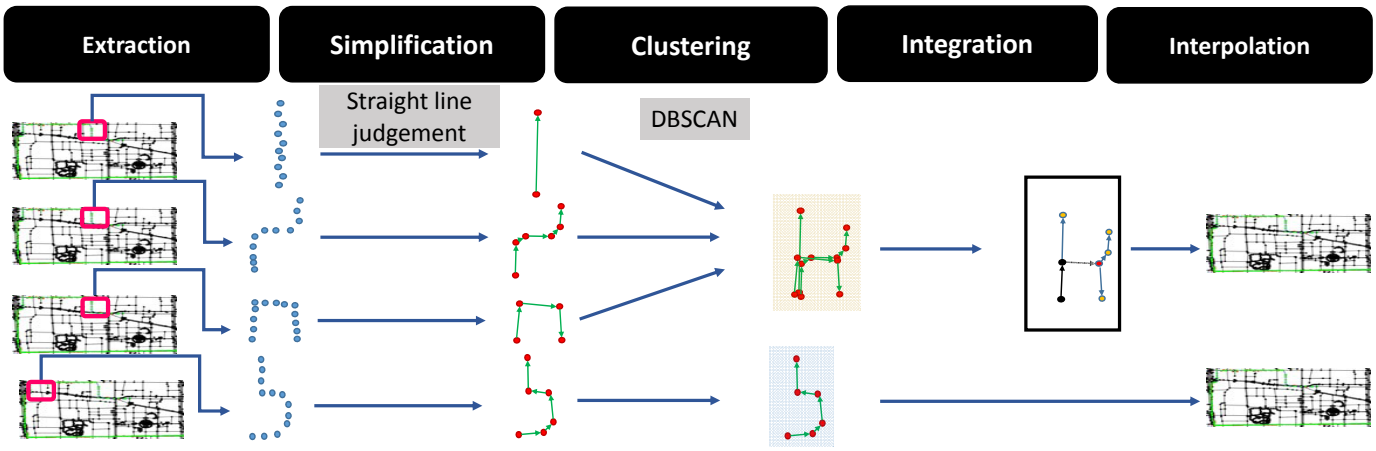



Fig. 6: An overview of our algorithm

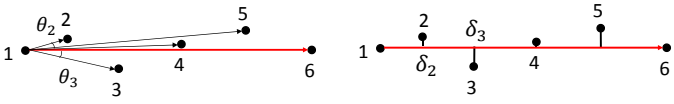



Fig. 7: An example of straight line judgment

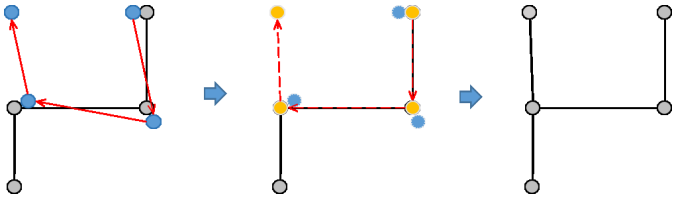


Fig. 8: An example of integration

### E. Clustering

If off-road trajectories are caused due to missing road segments, several off-road trajectories can be close each other. On the other hand, if off-road trajectories are caused due to GPS errors, the off-road trajectories are distant from other trajectories. Therefore, we find sets of trajectories that are close each other for reliable interpolation. Our clustering technique is based on DBSCAN [7]. First, we compute the representative point $l_t$ of an off-road trajectory by the following equation:

$$l_t = \textit{centroid of } p_1.l \textit{ and } p_2.l, \textit{ where } dist_E(p_1.l, p_2.l) \geq dist_E(p.l, p'.l) \textit{ for } \forall p, p' \in t. \quad (2)$$

$l_t$ is a centroid of two GPS points that have the longest Euclidean distance among GPS points in the off-road trajectory. We set $\mathbb{L}$ as the set of $l_t$ for each $t$.

We cluster the $\mathbb{L}$ for each off-line trajectory by DBSCAN. Here, the parameters of DBSCAN affect clustering results, and thus we have to set appropriate values for the parameters. DBSCAN has two parameters $\epsilon$ and $minPts$ to determine the density of points. We set a user-specified value to $minPts$ but compute $\epsilon$ from trajectories. $\epsilon$ is computed as the following equation.

$$\epsilon = \max_{t \in \mathbb{T}} \max_{p \in t} dist_E(l_t, p.l). \quad (3)$$

Intuitively, $\epsilon$ is the largest value among Euclidean distances between the centroid points and each GPS point on the trajectories corresponding to the centroid points. The reason we set $\epsilon$ by Equation 3 is that their distances between the road segments should be closer than $\epsilon$ if trajectories represent the same missing road segments. In other words, if the trajectories do not represent the same missing road segments, their distances should be larger than $\epsilon$. Meanwhile, as we mentioned above, $minPts$ is given by users in our algorithm. It is highly related to the accuracy and comprehensiveness of interpolation. When $minPts$ is large, we interpolate the missing road segments on which many vehicles pass, and the number of interpolated road segments becomes small. When $minPts$ is small, we interpolate many missing road segments with less accuracy. Thus, users set arbitrary values to $minPts$ depending on their requests (i.e., comprehensive or reliable interpolation).

### F. Integration and interpolation

Even if off-road trajectories belong to the same cluster, they may not represent the same road segment, for example crossroads. Since the trajectories with same cluster may represent multiple road segments, we need to distinguish different road segments and integrate the road segments by joining them by intersections.

For integrating off-road trajectories, we handle the trajectories as a graph $G_t$. First, we pick the off-road trajectory with the largest number of GPS points among the trajectories in the same cluster. The trajectory is denoted by $G_t = (V_t, E_t)$, where $V_t$ and $E_t$ represent the set of GPS points and sequences between the GPS points, respectively. For other trajectories, if their GPS points are not closer to any points in $V_t$ than $\sigma$, we add the points to $V_t$ and add the sequences to $E_t$. By this process, we obtain a graph that represents missing road segments. Since the first and end of GPS points of off-road trajectories are mapped to the points on road data, we naturally interpolate the road segment to the road data.

Figure 8 shows an example of integration. In this example, we integrate two trajectories (gray and blue points) after the simplification process. $G_t$ first represents the gray trajectory. Three GPS points of blue trajectories are close to the GPS points of the gray trajectory while the left-top GPS points is not closer to them. Thus, the left-top GPS point is added to $G_t$. Finally, we obtain the integrated road segments.

### G. Pseudo-code of our algorithm

Algorithm 1 shows the pseudo-code of our algorithm. It first extracts the set of trajectories that are through missing road segments with high probabilities (lines 1 – 5). Then, the extracted trajectories are simplified to take straight as possible (lines 6 – 15). Before executing DBSCAN, our algorithm computes the set of $\mathbb{L}$ representative points of off-line trajectories and $\epsilon$ (lines 16 – 28). It executes DBSCAN with $\mathbb{L}$, $\epsilon$, and the given $minPts$ (line 29). Finally, it integrates the off-line trajectories belonging to the same cluster and interpolates the integrated trajectories as missing road segments to the road data (lines 30 – 41).

**Algorithm 1:** Algorithm for Interpolating missing road segments

**input :** Set of trajectories $\mathbb{T}$, road data $R$, parameters; $\psi$, $\tau$, $\theta$, $\delta$, $minPts$, $\sigma$
**output:** Interpolated road data $R$

```
   /* Extraction                                                    */
1  T_o ← MapMatching(T, R);
2  C ← null;
3  for ∀t ∈ T_o do
4      if t.n < τ or d(t) < ψ then
5          remove t from T_o;
   /* Simplification                                                */
6  T_s ← null;
7  for ∀t ∈ T_o do
8      t_s ← {t.begin};
9      while t.size ≥ 2 do
10         t_tmp ← t;
11         while not StraightJudge(t_tmp, θ, δ) do
12             t_tmp.pop();
13         t_s ← t_s ∪ t_tmp.end;
14         t.remove(t_tmp.begin, t_tmp.end -1);
15     T_s ← T_s ∪ t_s;
   /* Clustering                                                    */
16 for ∀t ∈ T_s do
17     d_max ← 0;
18     for ∀p_1, p_2 ∈ t do
19         if d_max < dist_E(p_1.l, p_2.l) then
20             d_max ← dist_E(p_1.l, p_2.l);
21             l_t ← centroid of p_1.l and p_2.l;
22     L ← L ∪ l_t;
23 ε ← 0;
24 for ∀t ∈ T_s do
25     l_t ← the corresponding point of t in L;
26     for ∀p ∈ t do
27         if ε < dist_E(l_t, p.l) then
28             ε ← dist_E(l_t, p.l);
29 C ← DBSCAN(L, ε, minPts);
   /* Integration and Interpolation                                 */
30 for ∀C ∈ C do
31     G_t ← t with the largest size in C;
32     for ∀t ∈ C do
33         for ∀p ∈ t do
34             if p is not closer to any v in V_t than σ then
35                 V_t ← V_t ∪ p; E_t ← E_t ∪ (p, v_p); v_p ← p;
36                 Flag ← true;
37             else
38                 if Flag then E_t ← E_t ∪ (p, v_p);
39                 Flag ← false;
40                 v_p ← the closest v to p;
41     add G_t to R;
```

## V. Experimental study

We perform experiments to evaluate the effectiveness and accuracy of interpolation of our algorithm. All the algorithms are implemented in C++ and run on an Intel(R) Xeon(R) CPU E5620 @ 2.40GHz with 32.0 GB of RAM.

### A. Dataset

For the experiments, we use real trajectory data T-drive [15] and road data OpenSteetMap. The T-drive dataset includes GPS data obtained by taxis in Beijing. The dataset includes 10,357 taxis data from 2008 February second to eighth. Some GPS data have incorrect data such as missing time stamps, latitude, or longitude, and very long distance between two GPS points with close time stamps. Therefore, we clean such incorrect data. We also eliminate trajectories that are not in Beijing or have a few number of points. Finally, the number of taxis is 101. The minimum, maximum, and average numbers of GPS points on taxis are 6,746, 147,739, and 30,617, respectively. We extract Beijing road data from OpenStreetMap. OpenStreetMap provides us free road data that is manually updated by voluntary users but incomplete. We summarize the trajectory data and road data in Tables II and III, respectively.

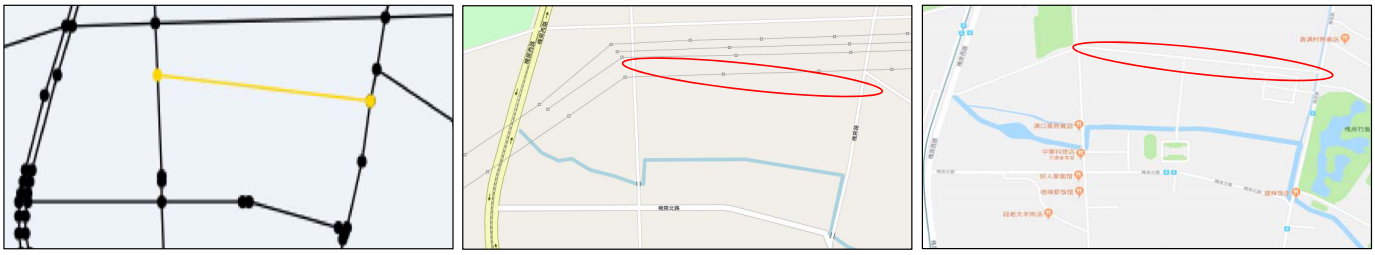
### B. Setting

Our algorithm interpolates missing road segments with several shapes. We categorize the shapes of interpolated road segments into three types; straight, curve, and complex (see Figures 9, 10, and 11). The straight type includes just two GPS points , and the other two types include more than two GPS points. The curve type includes no junctions, while the complex type includes junctions.

Our algorithm has several parameters. For the extraction step, we use 2 and 5 meters as $\tau$ and $\psi$, respectively. For the simplification step, we use 15 degrees and 2 meters as $\theta$ and $\delta$, respectively. For the integration step, we use 2 meters as $\sigma$. We vary $minPts$ for evaluating the accuracy of interpolation.

### C. Examples of interpolated road segments

We show the results of missing road segments interpolated by our algorithm. Figures 9, 10, and 11 show examples of



(a) Interplated road segments (b) OpenStreetMap (c) Google map

Fig. 9: Interpolating a straight road segment

(a) Interplated road segments (b) OpenStreetMap (c) Google map

Fig. 10: Interpolating a curve road segment

(a) Interplated road segments (b) OpenStreetMap (c) Google map

Fig. 11: Interpolating a complex road segment

TABLE II: Trajectory

| | |
|---|---|
| # of taxis | 101 |
| The min # GPS points per taxi | 6,746 |
| The max # GPS points per taxi | 147,739 |
| Average # GPS points per taxi | 30,617 |

TABLE III: Road data

| | |
|---|---|
| Date | 2015 June 27th |
| Location | Beijing |
| # of vertices | 306,311 |
| # of edges | 334,315 |

successful interpolation. These figures show (a) the interpolated road segments, (b) OpenStreetMap, and (c) Google map. We first compare the road data between OpenStreeMap and Google map. These figures show the OpenStreetMap in Beijing is incomplete compared to Google map. From these figures, we can see the importance of updating real-world road data.

Figure 9 shows the result of a straight segment. Our algorithm interpolates the road segment whose shapes is almost same to the road segment on Google map. Our simplification technique works well to interpolate straight road segments. Next, Figure 10 shows the result of a curve segment. The curve segment is also interpolated well by simplifying trajectories. Finally, Figure 11 shows the result of complex road segments. The complex road segment has several junctions. Our integration step integrates the trajectories for connecting them and avoiding redundant road segments. Our algorithm modifies the off-road segments to the realistic shapes and appropriately integrates the multiple road segments.

Our algorithm can interpolate several types of missing road segments. It is useful for updating the real-world incomplete road data.

TABLE IV: # of interpolated segments for each $minPts$

| $minPts$ | # of clusters | Straight | Curve | Complex |
|---|---|---|---|---|
| 1 | 512 | 109 | 102 | 301 |
| 2 | 167 | 18 | 19 | 130 |
| 3 | 75 | 5 | 5 | 65 |
| 4 | 48 | 3 | 3 | 42 |
| 5 | 37 | 2 | 2 | 33 |

TABLE V: Accuracy for each $minPts$

| $minPts$ | # of samples | Straight | Curve | Complex | Accuracy |
|---|---|---|---|---|---|
| 1 | 60 | 19/17 | 21/18 | 17/11 | 76.6% |
| 2 | 20 | 2/1 | 3/3 | 15/8 | 60.0% |
| 3 | 8 | 2/0 | 0/0 | 6/4 | 50.0% |
| 4 | 6 | 0/0 | 0/0 | 6/3 | 50.0% |
| 5 | 5 | 0/0 | 0/0 | 5/2 | 40.0% |

### D. Comprehensiveness and Accuracy

We evaluate the comprehensiveness and accuracy of our algorithm. A large number of interpolated road segments indicates high comprehensive interpolation.

Table IV shows the numbers of interpolated road segments for each $minPts$. From Table IV, we can see that the number of clusters decreases as $minPts$ increases. When $minPts$ is small in DBSCAN, the GPS points likely become core. On the other hand, when $minPts$ is large, the GPS points likely become noise. Comparing the number of interpolated road segments for each type, the number of complex road segments is larger than those of the other two types. Since interpolated road segments are mainly in residential areas, the road segments are often complex. For example, our algorithm detects parking areas as missing road segments. The reasons that interpolated segments are mainly in residential areas are that (1) we use the trajectories of taxis and (2) missing road segments are often minor roads.

We judge the interpolation is accurate if the missing road segments exist in Google map. Since the number of interpolated road segments is large, we randomly pick some interpolated road segments as samples for evaluating the accuracy. Table V shows the numbers of samples and the accuracy of interpolation for each type and $minPts$. In the table, $x/y$ denotes the number $x$ of samples and the number $y$ of accurate interpolated roads among samples. From this table, we can see that the accuracies of straight and curve road segments are higher than that of complex road segments. Since the two types are simpler than complex one, such interpolated road segments become accurate. As the number of $minPts$ increases, the number of complex road segments increases, and thus the accuracy decreases. This is because the interpolated road segments are often complex when $minPts$ is large, and it is difficult to appropriately interpolate the complex road segments. Improvement of the accuracy of the complex road segments is a part of our future work.

## VI. Conclusion

In this paper, we proposed an algorithm for interpolating missing road segments by using vehicle trajectory data. The algorithm uses map matching and clustering techniques to appropriately interpolate missing road segments. Through experiments using real incomplete road data and trajectory data, we verified that our algorithm effectively interpolates missing road segments.

As a part of our future work, we apply our algorithm to road data on developing countries where OpenStreetMap is more reliable than Google map such as Mongolia. We also employ machine-learning techniques to learn the shapes of the road segments from existing road data to more naturally interpolate road segments.

## Acknowledgement

This research is partially supported by the Grant-in-Aid for Scientific Research (A)(JP16H01722) and Grant-in-Aid for Young Scientists (B)(JP15K21069).

## References

[1] J. Biagioni and J. Eriksson, "Map inference in the face of noise and disparity," in *ACM SIGSPATIAL*, 2012, pp. 79–88.
[2] L. Cao and J. Krumm, "From GPS traces to a routable road map," in *ACM SIGSPATIAL*, 2009, pp. 3–12.
[3] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas, "City-scale map creation and updating using GPS collections," in *KDD*, 2016, pp. 1465–1474.
[4] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *ICDE*, 2011, pp. 900–911.
[5] J. J. Davies, A. R. Beresford, and A. Hopper, "Scalable, distributed, real-time map generation," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 47–54, 2006.
[6] S. Edelkamp and S. Schrödl, "Route planning and map inference with global positioning traces," *Computer science in perspective*, pp. 128–151, 2003.
[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.
[8] A. Fathi and J. Krumm, "Detecting road intersections from GPS traces," in *ACM SIGSPATIAL*, 2010, pp. 56–69.
[9] T. Guo, K. Iwamura, and M. Koga, "Towards high accuracy road maps generation from massive GPS traces data," in *Geoscience and Remote Sensing Symposium*, 2007, pp. 667–670.
[10] J.-H. Haunert and B. Budig, "An algorithm for map matching given incomplete road data," in *ACM SIGSPATIAL*, 2012, pp. 510–513.
[11] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *ACM SIGSPATIAL*, 2009, pp. 352–361.
[12] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *ACM SIGSPATIAL*. ACM, 2009, pp. 336–343.
[13] S. Rogers, P. Langley, and C. Wilson, "Mining GPS data to augment road models," in *KDD*, 1999, pp. 104–113.
[14] Y.-T. Wen, J. Yeo, W.-C. Peng, and S.-w. Hwang, "Efficient keyword-aware representative travel route recommendation," *IEEE TKDE*, vol. 29, no. 8, pp. 1639–1652, 2017.
[15] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *ACM SIGSPATIAL*, 2010, pp. 99–108.