Constructing Street Networks from GPS Trajectories*

Mahmuda Ahmed and Carola Wenk

Department of Computer Science
University of Texas at San Antonio
{mahmed,carola}@cs.utsa.edu

Abstract. We consider the problem of constructing street networks from geo-referenced trajectory data: Given a set of trajectories in the plane, compute a street network that represents all trajectories in the set. We present a simple and practical incremental algorithm that is based on partial matching of the trajectories to the graph. We use minimum-link paths to reduce the complexity of the reconstructed graph. We provide quality guarantees and experimental results based on both real and synthetic data. For the partial matching we introduce a new variant of partial Fréchet distance.

1 Introduction

We study the task of developing geometric algorithms with quality and performance guarantees for constructing street networks from geo-referenced trajectory data. This is a new type of geometric reconstruction problem in which the task is to extract the underlying geometric structure described by a set of movement-constrained trajectories, or in other words reconstruct a geometric domain that has been sampled with continuous curves that are subject to noise. As a finite sequence of time-stamped position samples, each input trajectory represents a finite noisy sample of a continuous curve.

Due to the ubiquitous availability of geo-referenced trajectory data, the road network construction task has widespread applications ranging from a variety of location-based services on street maps to the analysis of tracking data for hiking trail map generation or for studying social behavior in animals. The underlying movement constraints can be in the form of an explicit road network that the trajectories move in. In other applications the movement might be constrained by non-geometric reasons such as behavioral patterns of animals, or the geometric domain may be implicitly given such as by wind or sea currents enabling efficient flight routes for birds or swim routes for sea turtles. In this scenario the "street" network represents a common path structure described by the set of input trajectories. From a theoretical point of view the street network construction task poses a new class of geometric shape-handling problems dealing with sets of continuous curves that are subject to noise.

* This work has been supported by the National Science Foundation grant NSF CAREER CCF-0643597.

The problem of constructing digital maps from GPS traces has been considered in the Intelligent Transportation Systems and GIS communities [1,2,3,4], but the presented solutions are of a heuristic nature and do not provide quality guarantees. Recently the street network construction problem has received attention in the Computational Geometry community [5,6,7]. Chen et al. [5] reconstruct "good" portions of the edges (streets) and provide connectivities between these sections. They bound the complexity of the reconstructed graph and they guarantee a small directed Hausdorff distance between each original and corresponding reconstructed edge. Aanjaneya et al. [6] viewed street networks as metric graphs and they prove that their reconstructed structure is homeomorphic to the original street network. Their main focus is on computing an almost isometric space with lower complexity, therefore they focus on computing the combinatorial structure but they do not compute an explicit embedding of the edges or vertices. Ge et al. [7] employ a topological approach by modeling the reconstructed graph as a Reeb graph. They define an intrinsic function which respects the shape of the simplicial complex of a given unorganized data point set, and they provide partial theoretical guarantees that there is a one-to-one correspondence between cycles in the original graph and the reconstructed graph. All these approaches provide quality guarantees under certain assumptions on the original street network and the input trajectories, but they are based on sub-sampling the trajectory data and then working with an unorganized set of sample points and local neighborhood properties.

Our Contribution We present a simple and practical incremental algorithm that is based on partial matching of the trajectories to the graph. Our algorithm preserves the shape of the edges (streets) in the reconstructed graph by maintaining the continuous structure of the input curves. We provide quality guarantees based on the Fréchet distance, which is a distance measure suitable for comparing shapes of continuous curves. For the partial matching we introduce a new variant of partial Fréchet distance. We provide experimental results of our algorithm for real vehicle trajectory data as well as for generated data, and a statistical comparison between the original and reconstructed graph.

We prove that there is a one-to-one correspondence with bounded complexity between *well-separated* portions of the original and the reconstructed edges. However, giving quality guarantees for portions of the graph where edges come close together, in particular in regions around vertices, is a much more challenging task. We provide the first attempt at reconstructing vertex regions and providing quality guarantees, without assuming very clean and extremely densely sampled data. We reconstruct intersections as sets of vertices within bounded regions (*vertex regions*), where the size of each set is bounded by the degree and the region is bounded by the minimum incident angle of the streets at that intersection. We guarantee that if the vertices are sufficiently far apart so that the vertex regions do not overlap, then the vertices of each vertex region correspond to exactly one vertex in the original graph.

2 Problem Statement, Data Model & Assumptions

We model the *original* graph (street network) $G_o = (V_o, E_o)$ as an embedded undirected graph in $\mathbb{R}^2$. Each trajectory in the input curve set I is assumed to have sampled a connected sequence of edges in G_o(*street-path*). We model the error associated with each trajectory by a precision parameter ε. Given an input set I of polygonal curves in the plane and a precision parameter $\varepsilon > 0$, our goal is to compute an undirected *reconstructed* graph $G = (V, E)$ that represents all curves in the set. I.e., a *well-separated* portion of each edge in E_o corresponds to a sub-curve of an edge in E, and each vertex in V_o corresponds to a set of vertices in V.

We assume that V_o and V are sets of vertices with degree > 2 and each edge is represented as a polygonal curve. We assume that G_o is fully sampled i.e., for each street $\gamma \in E_o$ there is a sampled sub-curve in input curve set, I. We refer to each edge of G_o as a *street*.

An input trajectory is a finite sequence of time-stamped position samples, which represents a finite noisy sample of a continuous curve. Generally, the measurements of the position samples are only accurate within certain bounds (measurement error), and the movement transition in between position samples can be modeled with varying accuracies depending on the application (sampling error). We model trajectory data as piecewise linear curves, and we will in particular consider trajectories of vehicles driving on a street network. In this case, the input curves in fact sample an $\omega/2$-fattening of a street-path, where ω is the street width. The δ-fattening of a point set A is the Minkowski sum $A_\delta = A \oplus B(0, \delta)$, where $B(x, r)$ is the closed ball of radius r centered at x.

We work with a single precision parameter ε which captures the different kinds of noise as well as the street width. We use the Fréchet distance [8] to measure the similarity between the shape of an input curve and a street-path. For two planar curves $f, g : [0, 1] \to \mathbb{R}^2$, the Fréchet distance δ_f is defined as

$$\delta_F(f, g) = \inf_{\alpha,\beta:[0,1]\to[0,1]} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \qquad (1)$$

where α, β range over continuous and non-decreasing reparametrizations, and $\|.\|$ denotes the Euclidean norm.

To define the well-separability of streets, we make use of the following definition from [5].

Definition 1 (α-Good). *A point p on G is* α-good *if $B(p, \alpha\varepsilon) \cap G$ is a* 1*-ball that intersects the boundary of $B(p, \alpha\varepsilon)$ in two points. A point p is* α-bad *if it is not α-good. A curve β is α-good if all points on β are α-good.*

2.1 Assumptions

The correctness of our algorithm depends on the following assumptions that we make about the original graph and input data, see Figure 1.

1. **Assumptions on G_o:** (a) Each street has a well-separated portion which is 3ε-good. We refer to this as a ***good section.*** (b) If for two streets γ_1, γ_2 there are points $p_1 \in \gamma_1$ and $p_2 \in \gamma_2$ with distance $\leq 3\varepsilon$, then γ_1 and γ_2 must share a vertex v, and the sub-curves $\gamma_1[p_1, v]$ and $\gamma_2[p_2, v]$ have Fréchet distance $\leq 3\varepsilon$ and they are fully contained in $B(v_0, 3\varepsilon/\sin\alpha)$. Here $\alpha = \angle p_1 v p_2$ (see Figure 1b).

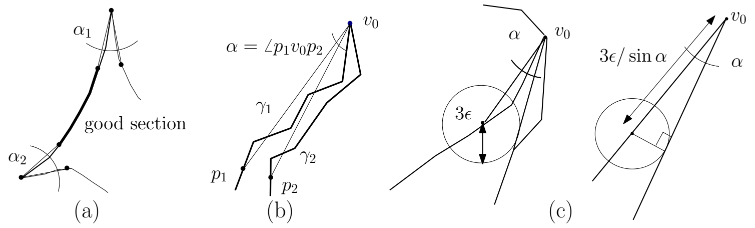


Fig. 1. (a) Assumption 1a (b) Assumption 1b (c) Assumptions yield minimum distance between two vertices.

From both assumptions follows that the minimum distance between two intersections is $> 3\varepsilon/\sin\alpha_1 + 3\varepsilon/\sin\alpha_2$. Assumption 1a states the minimum requirement to justify the existence of a street based on it being well-separated from other streets. Assumption 1b requires streets that are close together to converge to a vertex, and it discards streets that are close but do not share a vertex, because the input curves do not clearly distinguish between such streets. Note that the bound $3\varepsilon/\sin\alpha$ can be large for small values of α. So, the assumptions allow streets to be close together for a long time but restrict them to go far off once they are close to a vertex.

2. **Assumptions on Input Data:** (a) Each input curve is within Fréchet distance $\varepsilon/2$ of a street-path in G_o. (b) All input curves sample an acyclic path in G_o.

Assumption 2a ensures that the street-path and the corresponding input curve have to be similar. Assumption 2b ensures that, during an incremental construction algorithm, the first curve in I that represents a new edge does not sample a cycle. Our algorithm in fact only needs the unmatched portion (defined in Subsection 3.1) to sample an acyclic path. And even if it samples a cyclic path, such a cycle can be split in order to maintain this assumption.

Note that if a curve is well-sampled then the sampled curve is naturally within a bounded Fréchet distance of the original curve, which fulfils Assumption 2a. In particular, for our test data of vehicle trajectories, the GPS device error is generally bounded, and data from OpenStreetMap.org is generally sampled every second, and it captures every feature of the shape of the original street very well.

3 Algorithm

3.1 Preliminaries

In our algorithm we employ the concept of the *free space* F_ε and the *free space surface* FS_ε of one curve and a graph to identify clusters of sub-curves which sample the same street-path. For two planar curves $f, g : [0,1] \to \mathbb{R}^2$, and $\varepsilon > 0$, *free space* is defined as $F_\varepsilon(f,g) := \{(s,t) \in [0,1]^2 |\ \|f(s) - g(t)\| \leq \varepsilon\}$. The *free space diagram* $FD_\varepsilon(f,g)$ is the partition of $[0,1]^2$ into regions belonging or not belonging to $F_\varepsilon(f,g)$. The *free space surface* $FS_\varepsilon(G,l)$ for a graph $G = (V,E)$ and a curve l is a collection of free space diagrams $FD_\varepsilon(e,l)$ for all $e \in E$ glued together according to the adjacency information of G (see Figure 3).

In [8] it has been shown that $\delta_F(f,g) \leq \varepsilon$ if and only if there exists a curve within F_ε from the lower left corner to the upper right corner, which is monotone in both coordinates, see Figure 2 for an illustration. $FS_\varepsilon(G,l)$ could be disconnected if the graph G is not connected.

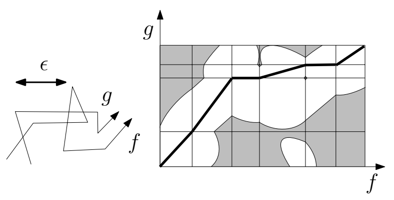


Fig. 2. FD_ε for two polygonal curves f, g. A monotone path is drawn in the free space.

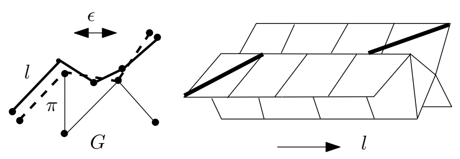


Fig. 3. FS_ε for a graph G and a curve l. An example path π is shown in dashed line and an l-monotone path in FS_ε is highlighted in bold.

In our algorithm we combine the idea of *map matching* and *partial curve matching* to map a curve partially to a graph. Buchin et al. [9] solved a *partial matching* for two curves by finding a monotone shortest path on a weighted FD_ε from lower left to upper right end point, where free space (white region) has weight 0 and non-free space (black region) has weight 1 (see Figure 4a). Alt et al. [10] solved the *map-matching* problem of matching a curve l to a graph by finding an l-monotone path in the free space of FS_ε from any left to any right end point. Here, the path is not allowed to penetrate the black region. In our case we need to find an l-monotone shortest path on the weighted free-space surface from any left end point to any right end point. However, finding such a shortest path on a weighted non-manifold surface is hard. Moreover, as the path can begin and end at any left or right end point, in some cases such a path does not provide us with the mapping we are looking for (see Figure 4a). The bold path is the desired one but the shortest path based on L_2 is the dashed one.

We used a minimum-link chain stabbing algorithm [11] to lower the complexity, such that the complexity of the representative edge will depend only on the complexity of the original street rather than on the complexity of the input curves.

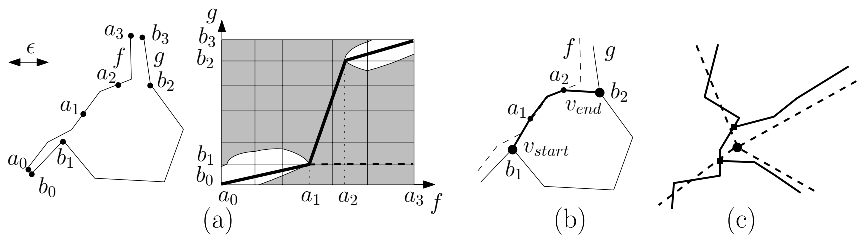


Fig. 4. (a) Partially similar f and g. $M_\varepsilon(f,g) = \{(f[a_0,a_1], g[b_0,b_1]), (f[a_1,a_2], null), (null, g[b_1,b_2]), (f[a_2,a_3], g[b_2,b_3])\}$. (b) Adding the unmatched portion of f as an edge. (c) G_o and G are in dashed and bold respectively.

3.2 Street Network Reconstruction Algorithm

Our algorithm can be divided into two phases, the first one involves computing a reconstructed graph (step 1, step 2) and in the second phase we compress the complexity of that graph (step 3). The simplicity of the algorithm relies on careful observation of how $FS_{1.5\varepsilon}(l,G)$ looks like when the original graph and the input curves satisfy the assumptions described in Subsection 2.1. In the first step it computes a partial curve-graph mapping.

Definition 2. *The Curve-Graph Partial Mapping* $M_\varepsilon(l,G)$ *consists of pairs* $(l[a_{j-1},a_j], X)$ *where* $0 = a_0 \le a_1 \le \ldots \le a_k = 1$ *such that every sub-curve* $l[a_{j-1},a_j]$ *in a partition* $l[a_0,a_1,\ldots,a_k]$ *of* l *is either mapped to a portion* $X = e[start,end]$ *of an edge* $e \in E$ *with* $\delta_F(l[a_{j-1},a_j], e[start,end]) \le \varepsilon$ *or* $X = null$ *if no such edge exists. A sub-curve is called a* matched portion *if it has a non-null mapping, and an* unmatched portion *otherwise. We assume that all unmatched portions as well as all matched portions along the same edge are maximal.*

Our iterative algorithm proceeds as follows: Let $G_i = (V_i, E_i)$ be the graph after the i-th iteration.

```
for each $l \in I$
    **Step 1:** Compute Curve-Graph Mapping $M_{1.5\varepsilon}(l, G_{i-1})$
    for each $(l[a_{j-1},a_j], X) \in M_{1.5\varepsilon}(l, G_{i-1}))$
    if $X$ is null
      // Add edge for the unmatched portion $l[a_{j-1},a_j]$
      **Step 2:** Create/split edges and create vertices
    else
      // Update $e[start,end]$ with matched portion $l[a_{j-1},a_j]$
      **Step 3:** Compute Minimum-Link Representative Edge
```

The total worst case runtime for adding the i-th curve is $O(N_{i-1}n_i+k_u(N_{i-1}+n_i)^2 \log(N_{i-1}+n_i)+k_c)$, where N_{i-1} is the complexity of G_{i-1}, n_i is the complexity of the ith curve $l \in I$, and k_u and k_c are the number of edges updated or created, respectively. The runtime for each individual step will be given with the description of the step below.

Step 1: Compute Curve-Graph Mapping $M_{1.5\varepsilon}(l_i, G_{i-1})$

In this step of the algorithm we compute a partial mapping $M_{1.5\varepsilon}(l_i, G_{i-1})$ which minimizes the total length of unmatched portions of l_i. First, we compute $FS_{1.5\varepsilon}(l_i, G_{i-1})$, and then we project the white regions onto the curve $(l\,[a_{j-1}, a_j]\,, e\,[start, end])$ which yields *white intervals* (matched portions). Lastly, we fill the gap between non-overlapping white intervals with a *black interval* $(l\,[a_{j-1}, a_j]\,, null)$ (unmatched portion).

In our setting, the objective is one-sided and we are interested in matching the curve maximally with the graph. Therefore the problem of computing the shortest monotone path reduces to computing the total length of the unmatched portions along the curve l. This allows the desired path to go through the black region along the graph direction without adding any additional cost. Therefore, we measure the length of the path **only within the unmatched portion and only along the curve direction**.

Our projection approach is justified by Assumption 1a on the street network that no two streets are less than or equal to 3ε close to each other if they do not share a vertex, which implies for each $l\,[a_{j-1}, a_j]$ if it samples a portion of a good section then only one $e\,[start, end]$ exists.

After this step we have the curve-graph mapping as a list of black and white intervals ordered by their start points on l_i. Such a list can be computed in $O(n_i N_{i-1})$ total time.

Step 2: Create/Split Edges and Create Vertices

In this step of the algorithm, for each $(l_i\,[a_{j-1}, a_j]\,, null) \in M_{1.5\varepsilon}(l_i, G_{i-1})$ we create an edge in constant time. By construction, the previous and next intervals of a black-interval are either *null* (the interval is the first or last element in the list) or white. Assume them to be non-*null* (if one or both of them are *null* then we do not have to create a vertex on the *null*-end). Let $e_{prev} = p_0p_1p_2 \ldots p_{n_1}$ and $e_{next} = n_0n_1n_2 \ldots n_{n_2}$ be the edges of previous and next intervals $(l_i\,[a_{j-2}, a_{j-1}]\,, e[start,\ end])$ and $(l_i\,[a_j, a_{j+1}]\,, e\,[start, end])$, respectively. Then $v_{start} = e_{prev}\,[end]$ and $v_{end} = e_{next}\,[start]$ are two points on the line segments $p_{i-1}p_i$ and $n_{j-1}n_j$. We create new points on the segments if $start > 0$ or $end < 1$, otherwise we take the existing endpoints as v_{start}, v_{end} and insert them into the vertex list as new vertices (multiple vertices might be created for a single original one, see Figure 4c). Then we follow the steps below to split an existing edge and create a new one. *a*) split e_{prev} as $p_0p_1 \ldots p_{i-1}v_{start}$ and $v_{start}p_i \ldots p_{n_1}$ *b*) split e_{next} as $n_0n_1 \ldots n_{j-1}v_{end}$ and $v_{end}n_j \ldots n_{n_2}$ *c*) insert $v_{start}l_i\,[a_{j-1}, a_j]\,v_{end}$ as an edge in E_{i-1}. For example, in Figure 4a, consider g

as an edge in G_{i-1} and f as l_i. Figure 4b shows the addition of the unmatched portion of f as an edge in the graph, here $e_{prev} = e_{next}$.

Step 3: Compute Minimum-Link Representative Edge

In first and second step we have computed the reconstructed graph for first i input curves. In this step, we compute a minimum-link representation for each $(l_i\left[a_{j-1}, a_j\right], X) \in M_{1.5\varepsilon}(l_i, G_{i-1})$, where $X \neq null$. The problem address the following: Given two polygonal curves f, g which both have bounded Fréchet distance to another polygonal curve γ such that $\delta_F(\gamma, f) \leq \varepsilon$ and $\delta_F(\gamma, g) \leq \varepsilon/2$, we have to find a minimum-link representative curve e of γ such that $\delta_F(\gamma, e) \leq \varepsilon$. First, we construct a combined vertex-sequence γ' of vertices of f and g by following a monotone mapping in $FD_{1.5\varepsilon}(f, g)$. According to Lemma 1, the polygonal curve associated with such sequence has Fréchet distance $\leq 2\varepsilon$ with γ. Then we apply the minimum-link algorithm [11] for the sequence of $B(v_{\gamma'}, 2\varepsilon)$ objects, with the additional restriction that the vertices must lie within $g^{\varepsilon/2}$, where $v_{\gamma'}$ are the vertices of γ'. The resulting path e obtained by our variant might not be the minimum-link path for the sequence of $B(v_{\gamma'}, 2\varepsilon)$, but the path has complexity less than or equal to the original curve γ, as $\gamma'^{2\varepsilon}$, f^{ε} and $g^{\varepsilon/2}$ all contain γ. Using the triangle inequality it can be proven that $\delta_F(e, \gamma) \leq \varepsilon$. In Step 3 of our algorithm, we update the edges of G_{i-1} for each white interval using the algorithm described above, where the polygonal curve f corresponds to $e\left[start, end\right]$ and g corresponds to $l_i\left[a_{j-1}, a_j\right]$. The time complexity to compute such a path is $O(n^2 \log n)$, where n is the number of vertices in γ'.

Lemma 1. *Let f and g be curves that sample a curve γ such that $\delta_F(f, \gamma) \leq \varepsilon$ and $\delta_F(g, \gamma) \leq \varepsilon/2$. Then any curve γ' comprised of vertices of f and g based on their order of a monotone mapping in $FD_{1.5\varepsilon}$ has $\delta_F(\gamma, \gamma') \leq 2\varepsilon$.*

4 Quality Analysis

In this section in Lemma 2 we prove that if an input curve $l \in I$ samples a good section of a street or a street-path, then that street-path is unique in G_o. It can be proven using a loop invariant that, if every edge in a path has exactly one good section, then after adding the i-th curve, the reconstruction graph G_i preserves all paths of ${G_o}^i$. By *preserving* a path we mean that all good sections have Fréchet distance less than or equal to ε to the original street and all vertices lie in the vertex region around the original vertices. Here, ${G_o}^i$ is the sub-graph of G_o fully sampled by the first i curves in I.

Lemma 2. *For each $l \in I$ there exists a mapping $M_{\varepsilon_p}(l, G_o) = \{(l\left[0, a_1\right], \gamma_1[b_0, 1]), (l\left[a_1, a_2\right], \gamma_2\left[0, 1\right]), \ldots (l\left[a_{k-1}, 1\right], \gamma_k\left[0, b_k\right])\}$ for $\varepsilon/2 \leq \varepsilon_p < 2.5\varepsilon$. And for $k \geq 3$ if l samples a good section of γ_1 and γ_k then $\gamma_1\gamma_2\gamma_3 \ldots \gamma_k$ otherwise $\gamma_2\gamma_3 \cdots \gamma_{k-1}$ is unique.*

4.1 Recovering good sections

In this subsection we prove that if a street $\gamma \in E_o$ is sampled by a set of input curves $I_\gamma = \{l_1, l_2, \ldots, l_k\}$, then our algorithm reconstructs each good section of γ as one edge $e \in E$.

Lemma 3. *For each edge $\gamma \in E_o$, if β is a good section of γ and there exists a non-empty set of input curves $I_\gamma = \{l_1, l_2, \ldots, l_k\} \subseteq I$ such that for every $l_i \in I_\gamma$, $\delta_F(l_i[start_i, end_i], \gamma) \leq \varepsilon/2$, then there exists only one $e \in E$ such that $\delta_F(e\,[start, end]\,, \beta) \leq \varepsilon$ and the complexity of $e\,[start, end]$ is less than or equal to the complexity of β.*

Proof. The proof follows from the construction of the curve-graph mapping $M_{1.5\varepsilon}(l_i, G_{i-1})$. When the first curve of the set is added to the graph, it is identified as an unmatched portion. And after that, all other sub-curves that sample γ will be identified as matched portions. Once we get the first matched portion (i.e., for the second curve in I_γ), we compute a center-line representative curve which ensures the minimum complexity and which has Fréchet distance $\leq \varepsilon$ from the original street. Thus, all the other sub-curves will also appear as matched portions in $M_{1.5\varepsilon}(l_i, G_{i-1})$, that means for all $l_i \in I_\gamma$ that sample β only one edge will be created in the graph. $\square$

4.2 Bounding Vertex Regions

In this section, we bound the vertex region, R_v for reconstructed vertices around the original vertex, v. We perform the analysis for a 3-way intersection which could easily be extended to an arbitrary n-way intersection.

Consider three streets γ_1, γ_2 and γ_3 incident to a vertex v_0. ${v_i}^{i-1}$ and ${v_i}^{(i+1)\%3}$, for $1 \leq i \leq 3$, are defined as the two points on γ_i which are farthest from v_0 along the curve (see Figure 5a). Here, $\angle {v_i}^{i+1} v_0 {v_{i+1}}^{i} = \alpha_i$ (see Figure 5b), and according to Assumption 2a, $v_0 {v_i}^{i+1}$ and $v_0 {v_{i+1}}^{i}$ are fully contained in $B(v_0, 3\varepsilon/\sin\alpha_i)$. For a 3-way intersection we can have input curves that sample three different street-paths and to reconstruct the vertex we need only two of them. Based on which two we use in which order, the reconstructed vertex can be in a different location in the vertex region. For analysis purpose, we define such a minimal input curve set, which is sufficient for reconstruction, considering all different choices. The idea is to compute the vertex region for each of these sets and then union them all to get R_{v_0}.

For a 3-way vertex we have six such sets. Let the first set I_1 contains l_i which samples street-path $\pi_1 = \gamma_1\gamma_2$ ($\delta_F(l_i, \pi_1) \leq \varepsilon/2$) and l_j which samples $\pi_2 = \gamma_2\gamma_3$ ($\delta_F(l_j, \pi_2) \leq \varepsilon/2$), where $i < j$. In the i-th iteration l_i will be inserted to G_{i-1} as an edge e, and in the j-th iteration l_j will be considered. To compute $M_{1.5\varepsilon}(l_j, G_{j-1})$, we compute intersection of ${l_j}^{1.5\varepsilon}$ and e and create a vertex with the intersection point which defines the mapping of a partition point of a matched and an unmatched portion of l_j. As e could be anywhere within ${\pi_1}^{\varepsilon}$ and l_j could be anywhere within ${\pi_2}^{\varepsilon/2}$. Considering all possibilities, we obtain the intersection region as ${\pi_1}^{\varepsilon} \cap {\pi_2}^{2\varepsilon}$. Again, as we are creating vertices only on the boundaries of the intersection region, no vertices would be created when the street-paths are $< 1.5\varepsilon$ close to each other, so the vertex region for I_1 is $R(v_0, I_1) = ({\pi_1}^{\varepsilon} \cap {\pi_2}^{2\varepsilon}) \setminus ({\pi_1}^{\varepsilon} \cap {\pi_2}^{\varepsilon})$ (see Figure 5c). Consolidating all six sets, the vertex region is shown in Figure 5d.

The above analysis can be extended to an arbitrary n-way intersection, where there are $n(n-1)/2$ possible paths and $r = (n(n-1)/2)!/(n(n-1)/2-(n-1))!$ sets of input curves are involved. The region is then defined as $R(v_0) = R(v_0, I_1) \cup R(v_0, I_2) \cup R(v_0, I_3) \cup \cdots \cup R(v_0, I_r)$.

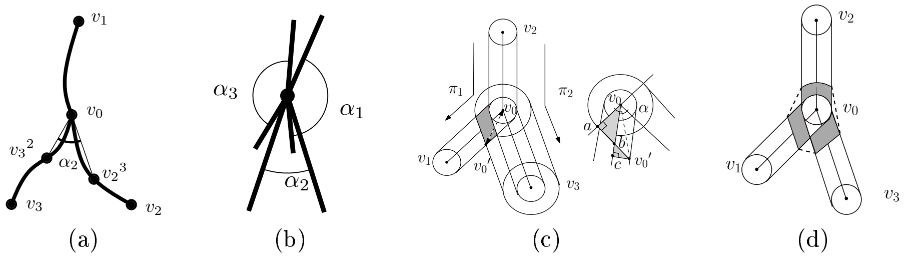


Fig. 5. (a) 3-way intersection. (b) 3-way intersection with illustration of α. (c) 3 way intersection region considering only I_1. (d) The shaded region is $R(v_0)$.

Lemma 4. *If $v_0' \in R(v_0)$ is the farthest point from a vertex $v_0 \in V_o$ with degree n, then $\varepsilon \le d(v_0, v_0') \le \varepsilon/\sin\alpha\sqrt{5+4\cos\alpha}$, where α is the minimum of all α_is.*

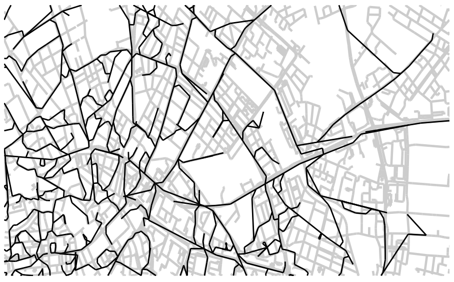
Proof. In Figure 5c, considering $\triangle v_0ab$ we have that $v_0b = v_0a/\cos(\pi/2-\alpha) = 2\varepsilon/\sin\alpha$. And considering $\triangle v_0'bc$ we have $bc = cv_0'/\tan\alpha = \varepsilon/\tan\alpha$. $d(v, v_0) = \sqrt{v_0c^2 + {v_0}'c^2} = \sqrt{\varepsilon^2 + ((2\varepsilon + \varepsilon\cos\alpha)/\sin\alpha)^2} = \sqrt{\varepsilon^2/\sin\alpha^2(5+4\cos\alpha)} = \varepsilon/\sin\alpha\sqrt{5+4\cos\alpha}$. □

5 Experimental Results

We implemented the first phase our algorithm using *java*. Instead of computing the minimum-link representative curve we used the unmatched portion of the first curve that defines an edge to represent the reconstructed edge. Experiments were conducted on a 2.93GHz Intel(R) Core(TM)2 Duo machine with 4.00GB of RAM. In this section we present our results obtained by running our algorithm on different datasets. It seems that, even using the noisy data our algorithm produces graphs with good accuracy.

We applied our algorithm on real tracking data obtained by sampling vehicle movements at a rate of 30 seconds. The dataset consists of 3, 237 vehicle trajectories consisting of a total of 57, 109 position samples. The data was collected from taxi cabs in the municipal area of Berlin, Germany, in 2007[1]. Figure 6

[1] A lot of tracking data is available at openstreetmap.org also, but it is hard to extract the desired type, since they contain a lot of hiking trails as well as vehicle trajectories.


(a)

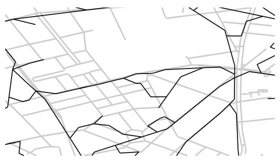

(b)

Fig. 6. Reconstructed graph of Berlin and a zoomed-in portion.

Table 1. Experimental Results

GPS	s.r.	ε	# of		G			G_o		
error	in sec	m	curves	points	edges	g.s.(m)	vertices	edges	g.s.(m)	vertices
± 5m	1	12	900	14515	805	105,062	56	560	76,248	37
	5	17	651	5934	731	66,755	46	560	67,959	32
	10	20	583	4318	681	56,379	32	560	65,270	28
	20	25	556	3323	733	62,522	28	560	58,810	26
	30	30	543	2793	696	65,784	22	560	53,200	20
± 10m	1	22	900	14515	912	76,991	47	560	63,128	28
	5	27	651	5934	501	46,031	31	560	56,251	24
	10	30	583	4318	499	31,816	31	560	53,200	20
	20	33	556	3323	605	37,709	20	560	49,643	17
	30	35	543	2793	635	50,112	16	560	48,183	14

shows the reconstructed graph (dark) overlayed with the original graph. As we can see, almost all sampled good sections were reconstructed, but as our data was noisy the distortion is high. For this case our ε was 82.5 meters and the running time was 11 minutes. The reconstructed graph had $3,044$ vertices and $3,546$ edges. Total complexity of all edges was $12,434$. As the original graph was not fully sampled it was hard to compare the graphs.

We generated our second type of dataset by almost fully sampling a subgraph of the Berlin graph which had 363 vertices and 560 edges/streets with a total length of $147,463$ meters. We generated multiple datasets by varying sampling rates (s.r.) and device error, which effectively influence the choice of ε. We assumed the speed of the vehicle is 88.5km/h. Our results are summarized in Table 1. The performance of our algorithm is measured in terms of number of vertices, number of edges and length of good sections (g.s.) in G and G_o. For simplicity of implementation we approximated the vertex region assuming $\alpha = 2\pi/7$, and mapped every reconstructed vertex to an original one which is within 3.5ε distance. As we can see, the number of reconstructed vertices in G is consistently higher than the number of good vertices in G_o. We identified good sections only with respect to vertices, and for each line segment we classified it as either good or bad. As we did not compute minimum-link representative edges, when the sampling rate is very small the length of the good section in reconstructed graph became larger than the original one (because the reconstructed edges zig-zag within very small distance). Our code is publicly available at `www.my.cs.utsa.edu/~mahmed/research`.

6 Conclusions

We outlined an incremental algorithm for street network construction and update. Our algorithm is very practical to use in real life as it is very simple to implement and it involves only one parameter ε. Unlike existing algorithms, it does not require very densely sampled data, as long as the features are captured, the algorithm produces pretty accurate maps in reasonable time. Our algorithm assures reconstruction of good sections and bounds the vertex regions. Our future work includes investigating vertex regions to reduce redundant vertices.

Acknowledgment We thank Dieter Pfoser and Maike Buchin for fruitful discussions.

References

1. Bruntrup, R., Edelkamp, S., Jabbar, S., Scholz, B.: Incremental map generation with GPS traces. In: Proc. IEEE Intelligent Transp. Systems. (2005) 574 – 579
2. Li, Z., Lee, J.G., Li, X., Han, J.: Incremental clustering for trajectories. In: DASFAA (2). (2010) 32–46
3. Guo, T., Iwamura, K., Koga, M.: Towards high accuracy road maps generation from massive GPS traces data. In: IEEE Int. Geoscience and Remote Sensing Symposium. (2007) 667 –670
4. Cao, L., Krumm, J.: From GPS traces to a routable road map. In: Proc. of the 17th ACM SIGSPATIAL Int. Conf. on Advances in GIS. GIS '09, New York, NY, USA, ACM (2009) 3–12
5. Chen, D., Guibas, L., Hershberger, J., Sun, J.: Road network reconstruction for organizing paths. In: Proc. ACM-SIAM Symp. on Discrete Algorithms. (2010)
6. Aanjaneya, M., Chazal, F., Chen, D., Glisse, M., Guibas, L.J., Morozov, D.: Metric graph reconstruction from noisy data. In: Proc. ACM Symp.Computational Geometry. (2011) 37–46
7. Ge, X., Safa, I., Belkin, M., Wang, Y.: Data skeletonization via Reeb graphs. In: 25th Annual Conference on Neural Info. Processing Systems. (2011) 837–845
8. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. Int. J. of Computational Geometry and Applications **5** (1995) 75–91
9. Buchin, K., Buchin, M., Wang, Y.: Exact algorithm for partial curve matching via the Fréchet distance. In: Proc. ACM-SIAM Symp. on Discrete Algo. (SODA09). (2009) 645–654
10. Alt, H., Efrat, A., Rote, G., Wenk, C.: Matching planar maps. Journal of Algorithms (2003) 262–283
11. Guibas, L.J., Hershberger, J.E., Mitchell, J.S.B., Snoeyink, J.S.: Approximating polygons and subdivisions with minimum-link paths. Int. J. of Computational Geometry and Applications (1993) 3–4