Clustering Trajectories for Map Construction

Kevin Buchin[*]
k.a.buchin@tue.nl

Maike Buchin[†]
maike.buchin@rub.de

David Duran[‡]
david.duran.rosich@est.fib.upc.edu

Brittany Terese Fasy[§]
brittany@cs.montana.edu

Roel Jacobs[*]
r.jacobs@student.tue.nl

Vera Sacristan[‡]
vera.sacristan@upc.edu

Rodrigo I. Silveira[‡]
rodrigo.silveira@upc.edu

Frank Staals[¶]
f.staals@uu.nl

Carola Wenk[‖]
cwenk@tulane.edu

[*] Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
[†] Faculty of Mathematics, Ruhr University Bochum, Germany
[‡] Dept. de Matemàtiques, Universitat Politècnica de Catalunya & BGSMath, Spain
[§] Computer Science Department, Montana State University, USA
[¶] Dept. of Information and Computing Sciences, Utrecht University, The Netherlands
[‖] Dept. of Computer Science, Tulane University, USA

ABSTRACT

We propose a new approach for constructing the underlying map from trajectory data. Our algorithm is based on the idea that road segments can be identified as stable subtrajectory clusters in the data. For this, we consider how subtrajectory clusters evolve for varying distance values, and choose stable values for these. In doing so we avoid a global proximity parameter. Within trajectory clusters, we choose representatives, which are combined to form the map. We experimentally evaluate our algorithm on vehicle and hiking tracking data. These experiments demonstrate that our approach can naturally separate roads that run close to each other and can deal with outliers in the data, two issues that are notoriously difficult in road network reconstruction.

CCS Concepts

•Information systems → Geographic information systems; •Theory of computation → Computational geometry;

Keywords

Trajectories, map construction, clustering, geometric algorithms

1. INTRODUCTION

Technology–in particular, the global positioning system (GPS)–has made tracking the location of a large number of moving entities feasible. As a result, vast amounts of trajectory data exists today. In a large proportion of this trajectory data, the moving entity is restricted to move on an underlying network. Consider, e.g., a car driving on roads, or a person walking along hiking trails. Recent research aims to reconstruct this underlying network from the trajectory data to aid in the construction of maps and detection of changes in the underlying networks; see [3, 4] for surveys. This, however, has turned out to be a challenging task.

One of the main problems is to determine to which candidate edge, road or path in the network, a subtrajectory corresponds to. Many of the proposed approaches deal with this by introducing a global spatial parameter modeling the road width; when the subtrajectory is within some distance ε of a candidate edge, it is assigned to this edge. The problem is, however, that in real data sets, no global value ε accurately captures the road width of all roads in the network [17]. For some road segments, we may need a much larger distance threshold than for others. For example, if the road or path is in a canyon, and GPS reception is poor, the trajectories deviate more from the actual underling road segment, or simply if a highway is much wider than a back-alley in the center of a city. Even if, after much manual parameter tweaking, the algorithm can be configured to produce a reasonably accurate map on a given data set, the same parameter values often do not give the best map on a different data set. Hence, more manual work is again required.

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-5490-5/17/11...$15.00
DOI: https://doi.org/10.1145/3139958.3139964

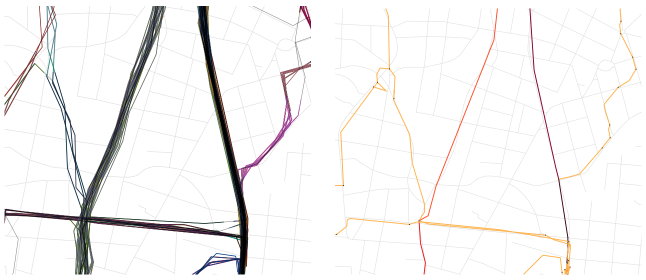
Figure 1: Example of bundle detection, with different colors indicating different bundles (left) and map construction (right) of part of the Athens data set.

We propose a new approach for map construction with the aim to alleviate this problem. In particular, our approach does not require a global parameter to model the road width. Instead, we find the desired road width for each edge in the network from the trajectory data itself.

More specifically, our algorithm consists of two phases: First, the algorithm clusters input (sub)trajectories into what we call *bundles*, or groups of entities (trajectories) that are close together for a sufficiently long time. Each such bundle may have a different size, length, and spatial "proximity". The algorithm finds appropriate values for these parameters by considering when the size and length of a bundle change significantly as a function of the spatial parameter ε. Second, the algorithm constructs the actual *map* (road network) from the bundles. Here, we use a simple greedy approach that constructs the map, starting from the largest bundle.

Our main contribution is the clustering of the trajectories into bundles and automatically selecting relevant bundles. This allows our algorithm to successfully reconstruct network edges of various widths and under noise for both vehicle and hiking tracking data. The algorithm succeeds in separating close, parallel roads under moderate noise. Based on subtrajectory clustering rather than point clustering, our algorithm can also handle data of varying sampling rates. See Figure 1 for an example.

Related Work. Several different approaches have been proposed in the literature for constructing street maps from trajectory data; see Ahmed et al. [3, 4] for an overview.

Point clustering algorithms [18, 21, 32] consider the input as a point cloud and then cluster these points in different ways to obtain intersections or street segments. For example, several algorithms use neighborhood complexes for the input point set [1, 12, 14, 20]. *Density-based* algorithms [2, 7, 13, 15, 25, 28, 29] first compute a density function over the set of input points, and then extract a roadmap from ridges in this density. The recent algorithm by Wang et al. [31] utilizes discrete Morse theory to construct a high quality map. These algorithms, however, generally do not fully exploit the continuous traversal information contained in the input trajectories. *Incremental track insertion* algorithms [5, 10, 26, 27] insert the trajectories one at a time into an initially empty map, often making use of map-matching ideas. In these algorithms, the order in which the trajectories are added may affect the resulting map. *Intersection linking* algorithms [19, 23] follow a two-step approach, by first detecting intersection nodes and in a second step using the trajectory data to connect the intersections. Karagiorgou and Pfoser's algorithm [23] identifies intersection nodes by detecting changes in movement (speed and direction) and clustering the resulting locations. Trajectory portions between locations in corresponding intersection nodes are then bundled together to form edges.

Our concept of bundles builds on the subtrajectory clustering algorithm of Buchin et al. [8]. There are various other notions and algorithms for subtrajectory clustering, e.g., the TRACLUS framework [24] and its many variants.

Problem Statement and Organization. Let $\mathcal{T}$ be a set of *trajectories*, that is, sequences of time-stamped locations. We assume that the entities that generated these trajectories moved on a fixed network, represented by an embedded graph G. Given the trajectories $\mathcal{T}$, we wish to reconstruct the part of G that the trajectories in $\mathcal{T}$ used.

In Section 2, we present our map-construction algorithm. First we show how to compute relevant clusters of subtrajectories, called *bundles*, in $\mathcal{T}$, then we show how to combine the clusters to construct a network. In Section 3, we evaluate our algorithm on vehicle and hiking tracking data.

2. ALGORITHM

Our algorithm consists of two main phases: extracting relevant bundles and constructing the network from these bundles. In the first phase (Section 2.1), we cluster the (sub)trajectories into bundles, and extract the most relevant bundles. We say that a set B is a bundle on $\mathcal{T}$, if it is a set of similar subtrajectories from trajectories in $\mathcal{T}$. If subtrajectories come from the same trajectory, they should not overlap in time (except possibly at their start/end points).

We consider bundles parameterized by their size, their length, and their spatial proximity. Specifically, a (k, ℓ, ε)-*bundle*, is a bundle with at least k subtrajectories, of which the longest subtrajectory has length ℓ and the pairwise distance between the subtrajectories is at most ε. As distance between subtrajectories, we use the Fréchet distance [6]. In our implementation, we use a variant that requires monotonicity only on vertices, not within segments [30]. In the colloquial Fréchet distance man-dog metaphor, they are allowed to backtrack but only within a segment. We choose this variant because it is easy and fast to compute.

Our goal is to reconstruct the map irrespective of direction, and for this, we allow traversing subtrajectories in either direction. To take direction into account, one could determine the direction of edges in a post-processing step. Alternatively, one could take directions into account already when computing the bundles. This, however, only makes sense if there is sufficient data to form bundles in both directions. Also, it would require an extra step of identifying edges of the same road in opposite directions.

We consider bundles for increasing distance parameter ε and select those that are relatively stable with respect to ε, as well as maximal in size and sufficiently large. We realize that *stay points*, subtrajectories that remain within a small area for a relatively long time, can create artificial bundles. To avoid this, the data should be pre-processed to remove stay points, which can be achieved by either explicitly removing stay points or by simplification. The latter will also reduce the size of the data and hence speed up calculation.

In the second phase (Section 2.3), we combine the relevant bundles to construct the network G. Each bundle has a representative subtrajectory with low distance to all the subtrajectories in the bundle. The algorithm stitches together the representatives of relevant bundles to obtain the network.

2.1 Relevant Bundles

Recall that a (k, ℓ, ε)-bundle is a cluster of size at least k, length ℓ, and distance at most ε of similar subtrajectories. Clearly, every subset of trajectories forms a (k, ℓ, ε)-bundle, for some, or even several, combinations of parameter values. Not all of these bundles are useful to construct a map; instead, we want to consider only "relevant" bundles. Next, we consider three properties that relevant bundles should have, that is to be *maximal*, *stable* and *large*, and describe a way to obtain such bundles. Also, we require that all subtrajectories within a (k, ℓ, ε)-bundle have length at least ε or are the complete trajectory. This avoids short artificial bundles and ensures that for large ε we have one maximal bundle.

Maximal bundles. Consider k parallel and horizontal line segments of length ℓ, spaced ε apart. Any subset of $k' < k$ trajectories (line segments) forms a (k', ℓ, ε)-bundle. In fact, a (k', ℓ', ε)-bundle can be found for any length $\ell' \leq \ell$ by looking at sub-trajectories. However, in this example, the only "relevant" bundle seems to be the one consisting of all k trajectories, and length ℓ. To capture this, we introduce subbundles and maximal bundles.

We say that a bundle B_1 is a *subbundle* of B_2 if every trajectory $T_1 \in B_1$ is the subtrajectory of a trajectory T_2 in B_2. We also say that B_2 *dominates* B_1. For example, the blue bundle in Figure 2 (left) is a subbundle of the red bundle.

We note that bundle B_1 could fail to be a subbundle of B_2 by just a small margin, for example, as in Figure 2 (right). Such a small margin may not be relevant; hence, we consider B_1 as an "approximate" subbundle of B_2. More formally, we say that T_1 is a λ-*subtrajectory* of T_2, if and only if

$$\operatorname{len}(T_1) \leq \operatorname{len}(T_1 \cap T_2) + \lambda,$$

where $\operatorname{len}(T)$ denotes the length of (sub)trajectory T. Hence, only a section of T_1 of length λ is not part of T_2. We extend this notion to λ-*dominates* and λ-*subbundle*. We now consider only (λ-)*maximal* bundles: bundles that are not a λ-subbundle of another bundle.

Note that λ governs the allowed distance between the endpoints of the trajectories involved. Hence, letting λ be a function of ε is a natural choice, so we simply set $\lambda = c\varepsilon$, for some small constant c, when comparing bundles for the same ε. We use $c = 2$ in our implementation; this essentially allows an error of ε on both sides of the bundle. For bundles of different ε by transitivity of (λ-)subbundles the larger ε is used as error margin.

Stable bundles. We consider only bundles that are "relatively" stable, with respect to variation in ε. That is, we have to increase ε by a large amount before new trajectories are added to the bundle. We give a more formal definition below. First, we note that bundles are monotone in ε in the sense that any (sub)trajectory that is present for parameter value ε will also be present as (sub)trajectory for any parameter value $\varepsilon' \geq \varepsilon$. Note that this still holds when we require a minimum length ε of the subtrajectories within a bundle, because the length of (sub)trajectories in the bundle will increase as much as ε up to their total length.

The motivation behind using stability as a measure of relevance is that we expect stable bundles to be "complete", meaning that no additional trajectories should be added to them unless we drastically change scale. With respect to map construction, such stable bundles may represent roads for the current scale. These bundles should be more important than bundles that are "incomplete" and still could collect additional trajectories at the current scale by slightly increasing ε. Note that each bundle (set of trajectories) may

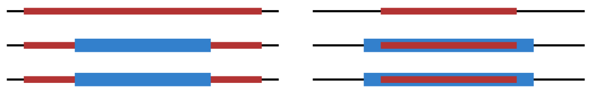
Figure 2: Examples of subbundles. Left: the red bundle completely covers the blue bundle; thus, the blue bundle is a subbundle of the red bundle. Right: the red bundle covers the blue bundle, except for a small margin in length. Here, the blue bundle is a λ-subbundle of the red bundle.

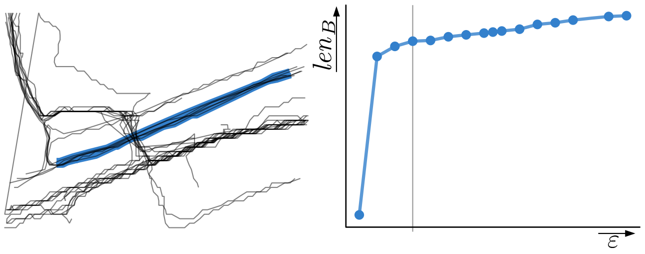

Figure 3: A bundle B and its length len_B as a function of ε.

have its own value(s) of ε for which it is stable, as each road may have its own road width.

Fix a set of trajectories B and consider the length of the bundle formed by the entities in B as a function of ε. More formally, let $len_B(\varepsilon) = \ell$ if and only if $\ell > 0$ is the largest value for which the entities in B form a $(|B|, \ell, \varepsilon)$-bundle. This function is monotonically increasing in ε. See Figure 3 for an example. In practice, we compute bundles for a discretized set of ε-values, up to some large upper bound on the scale at which we consider bundles (in our experiments, we use 100m for the hiking data and 200m for the vehicle data).

The trajectories in B now form a maximal bundle on an interval $I = [\varepsilon, \varepsilon']$. The interval starts at a value ε at which B is born as a maximal bundle, and ends at a value ε' for which B becomes a sub-bundle of another bundle (and hence dies as a maximal bundle). We refer to the length of this interval as the *lifespan* of (this instance of) bundle B. Furthermore, we define $(\varepsilon' - \varepsilon)/\varepsilon = \varepsilon'/\varepsilon - 1$ as the *relative lifespan* of (this instance of) bundle B. We use relative lifespan rather than absolute lifespan, since we expect a bundle that lives for instance between $\varepsilon = 5$ and $\varepsilon = 10$ to be more relevant than one that lives between $\varepsilon = 30$ and $\varepsilon = 35$.

We consider each maximal bundle (interval) that has a large relative lifespan (larger than 1) to be stable: we have to increase ε significantly before the set of trajectories in the bundle changes. Next, we need to determine the value of ε associated with such a maximal stable bundle, which also determines the exact length of the bundle. Intuitively, we pick ε minimal such that the bundle obtained its (close to) maximal length. For this, we again use the function len_B. For instance, in Figure 3, we would pick a ε at the indicated value at which the bundle has approximately reached its final length. Also, we require an absolute minimum lifespan L for the bundles to get rid of noise in the input data.

Large bundles. A final property that we want from our bundles is that they are large enough, since we want to have a sufficient number of "witnesses" before adding an edge to the network. Since smaller bundles do not influence larger bundles, we do not explicitly avoid them when constructing the network, but simply filter out the corresponding network edges in a postprocessing step. (In our experiments, we removed edges with only one or two subtrajectories.)

2.2 Generating bundles

To implement our algorithm, we discretize the range of ε-values by picking equally spaced values $\varepsilon_1, \varepsilon_2, \ldots$ (in our experiments, we use multiples of 5m). For each such value ε_i, we fix generate all maximal length bundles of size k (and distance parameter ε_i) using a slightly modified version of the subtrajectory clustering algorithm by Buchin et al. [8].

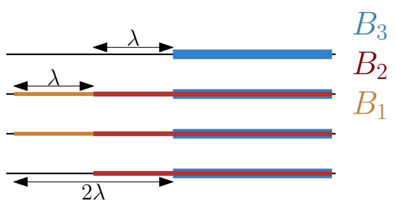

Figure 4: B_1 is a λ-subbundle of B_2, and B_2 is a λ-subbundle of B_3. However, B_1 is not a λ-subbundle of B_3.

The clustering algorithm [8] detects clusters (bundles) of similar subtrajectories, where all subtrajectories within a bundle have at most a given distance ε to a *representative subtrajectory* within the cluster. The algorithm requires either a minimal size or length of a cluster and optimizes for the other. The algorithm detects clusters in the *free space diagram*, the geometric data structure used to compute the Fréchet distance [6]. It concatenates all input trajectories to one trajectory T and uses the free space diagram of T with itself. Its main insight is that a cluster of size k with representative subtrajectory from vertex i to j of T can be detected in the free space as k non-overlapping monotone curves between vertices i and j. To detect these the algorithm sweeps two vertical lines over the free space, searching for such k non-overlapping monotone curves. The right sweep line is moved as far as possible as long as k curves exists. If these no longer exist the left sweep line is moved towards the right one.

We modify this algorithm in order to find all maximal length bundles rather than a (single) maximum length bundle, and to detect only bundles for which also the shortest trajectory is sufficiently long. To report all maximal bundles, we simply report a bundle every time before the left sweep line is moved. The clustering algorithm builds a labeled graph for searching between the sweep lines.

Rather than fixing a value for the size k, we start with $k = 1$ and increase by one until no more bundles are found. Once we have generated all bundles $\mathcal{B}_i$ for ε_i, we remove all bundles that are λ-dominated by another bundle in $\mathcal{B}_i$. Hence, we obtain a set of bundles that are maximal with respect to the other bundles in this set.

Since the λ-subbundle (and thus the λ-dominated) relation is not transitive, the order in which we remove subbundles matters; see Figure 4. We consider the bundles in order of decreasing size (and decreasing length in case of ties). This makes sure that we remove subbundles only if they are a subbundle of a bundle that is selected in the set.

Once we have generated the sets of maximal bundles for all ε_i, we link bundles for different ε_i in order to determine their lifespan. To do so, we determine for each fixed ε_i, which maximal bundles are a continuation of a bundle for ε_{i-1}, or arise from merging of bundles for ε_{i-1}, or are new bundles. With this information we can compute the lifespan of each bundle [22].

Parameters. Our approach uses one main distance parameter ε, the distance of subtrajectories within a bundle. However, this parameter is used only internally, i.e., it does not need to be set by the user. Instead, it is varied to detect bundles at different scales (distances). As external parameters we use only $\lambda = c\varepsilon$ as error margin for subbundles, where c is a small constant (which we pick as $c = 2$), and some simple thresholds on the minimum size and lifetime of bundles. This avoids picking up small noise as bundles. So our approach is essentially free of distance related parameters.

2.3 Constructing the Map

Once we have found all stable maximal bundles we use an incremental greedy algorithm to construct the map. We start from an empty map and consider the bundles in order of decreasing size k (recall that k is the number of trajectories represented in the bundle), adding them as edges to the map in order. In addition to adding edges to the map, every time we add a bundle B, we update all the others that overlap with B. Next, we elaborate on these two steps.

Each bundle B contains a set of subtrajectories that match to a common path in the (unknown) network. Recall that when the bundles are generated, each bundle is assigned a representative subtrajectory that represents this path. Let $p_{\text{start}}(B)$ and $p_{\text{end}}(B)$ be the endpoints of the representative subtrajectory of bundle B.

A bundle can overlap with other bundles, see for example Figure 5. In particular, larger bundles often overlap with longer ones that contain fewer subtrajectories. Therefore, when we add a bundle B we remove the subtrajectories that form B from the remaining bundles. Let B' be a bundle that has subtrajectories that share parts with those of B. We update B' as follows.

We subtract from B' the parts of the subtrajectories in common with B, possibly shortening B'. Every time a bundle B' gets shortened, its endpoints $p_{\text{start}}(B')$ and $p_{\text{end}}(B')$ are updated by connecting the end of the representative subtrajectory of B' to the closest of $p_{\text{start}}(B)$ and $p_{\text{end}}(B)$, using a line segment. This is done to guarantee that connectivity is preserved. Furthermore, if one of the subtrajectories in common was the representative of B, we make it also representative for B'.

It can also happen that when the parts in common in B and B' gets substracted from B', B' becomes disconnected. In this case, we replace B' by two new bundles B'_1 and B'_2, and proceed similarly to before.

Figure 5 shows an example with three bundles, one in red, one in blue, and one comprised of red and blue subtrajectories (highlighted in purple). The edge for the purple bundle is added first, breaking the red and blue bundles into two new bundles each. The edges for these new bundles connect to the endpoints of the edge of the purple bundle, as shown by the dashes.

Finally, it remains to consider the case of having to update a bundle B' that only partially overlaps B, that is, there is some trajectory in B that has no intersection with B', and vice versa. In this case, we preserve connectivity by adding all such trajectories to the bundles that get shortened or to the new bundles. For more details on this, we refer the reader to [22].

3. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of our algorithm. We base our evaluation on two recent cross-comparisons of map construction algorithms [3, 17]. The main difference between both comparisons is that the first one is based on urban vehicle data, while the second one uses hiking trajectories.

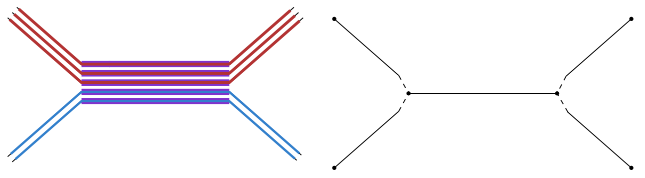

Figure 5: Three sets of stable bundles, blue, red, and purple, (left), where the purple bundle overlaps the red and blue, and the resulting map (right).

For the vehicular data, we use two data sets from [3], namely *Athens-small* and a randomly selected subset of trajectories of *Chicago*. The data set Athens-small has 129 trajectories and –after preprocessing, see below– 1955 vertices. The subset of the Chicago consists of 140 trajectories with a total of 1716 vertices (after preprocessing). Since the goal of the experiments is to evaluate the quality of the results obtained by the bundling approach, we have not yet optimized our implementation for scalability and therefore do not test it on the complete Chicago data set.

For the hiking data, we use subsets of the four data sets from [17]. They consist of trajectory data obtained from the trajectory-sharing website `Wikiloc.com` from four different areas in Catalonia, Spain. Two areas, *Delta* and *Aiguamolls*, are rather flat, while the other two, *Garraf* and *Montseny*, cover parts of mountain ranges. The *Delta* and *Aiguamolls* are mostly agricultural lands with little vegetation, while *Garraf* is mainly shrubland, and *Montseny* is covered by dense forest. In our experiments, we chose subsets of the different hiking data sets that contain interesting or difficult cases. The subsets had between 20 and 80 trajectories, and consisted of at most 2261 vertices (after preprocessing).

Preprocessing. For both the vehicular and the hiking data, we use a simple line simplification algorithm [16] to remove stay points and further simplify the input trajectories. For the vehicular data, the allowed error in the simplification is set to 10 meter, and for the hiking data to 5 meter.

3.1 Urban Road Networks

In this section we describe the results obtained with the two urban data sets, and comment on some interesting features of the generated maps.

Athens. Figures 6a and 6b show the bundles and the resulting map. The result has several noteworthy features. First, the algorithm is able to properly deal with different densities in trajectories and road widths. In Figure 6c, we have a large road with 59 trajectories, of which six branch off to a smaller road. Both roads are properly represented in the map. Only two out of the seven algorithms evaluated in [3] produced comparable results in this situation.

Secondly, by having a proximity parameter for each bundle, the algorithm is able to identify intersections as locations where bundles meet, represent intersections even if the trajectories within the bundles are more spread apart. An example of this is shown in Figure 6d. By having appropriate values for the proximity parameter we have bundles for the different directions, which allows us to have the intersection represented. Only one out of the seven algorithms evaluated in [3] produced comparable results in this situation.

Chicago. Figures 7a and 7b show the bundles and resulting map for the Chicago data set. We note that the algorithm is able to deal with trajectories that still stay relatively close to the actual road, but show clearly noisy parts, see for example Figures 7c and the vertical edge in 7d. For very noisy data, for example the horizontal trajectories at the bottom of Figure 7d, multiple small bundles are detected.

3.2 Hiking Networks

In the following paragraphs, we briefly discuss how our algorithm deals the main issues reported in [17] that concern the behavior of previous map construction algorithms for hiking data. The issues are illustrated in Figure 8.

Noisy trajectories. Noise is one of the most important issues that a map construction algorithm must be able to handle well. A frequent but difficult situation arises when multiple trajectories follow the same path, but the error causes the trajectories to be displaced around the actual path, being relatively far from each other. In hiking data sets with narrow paths through areas with poor GPS reception, like dense forest, this occurs rather often; see Figure 8a. For our algorithm, widespread trajectories are not a problem if we allow the bundle-specific ε to be large enough to cover them all; see Figure 9.

Zig-zags, sharp turns and bifurcations. Zig-zagging paths are one of the most challenging situations in hiking data, as witnessed for all algorithms tested in [17]. In particular, proximity thresholds (common to many network reconstruction algorithms) do not work well with zig-zags. Most algorithms either tend to add a large number of shortcuts or to merge the paths close to the turns, as for example in Figure 8b. In contrast, our algorithm uses bundles to reconstruct the different paths, and each bundle is represented by a part of an input trajectory, so the geometry of a zig-zag or winding path is mostly reconstructed; see Figure 10.

Generally, sharp turns cause issues for many algorithms. For instance, in Figure 8c the parts of the path before and after a turn were collapsed into one. However, sharp turns are not unusual in hiking data. Typically, they occur as part of a zig-zag along a path. As shown in Figures 10 and 11, our algorithm succeeds in detecting and retaining such zigzags, rather than collapsing them into a single edge, as is done by some of the other algorithms.

Similarly, many previous algorithms have trouble with bifurcations (see Figure 8d). Our algorithm has no problem in dealing with this situation (see Figure 12).

Irregular sampling rate. The heterogeneous hiking data sources used in [17] resulted in very irregular sampling rates, something that causes problems for many algorithms. One such problem is that two parallel trajectories, even at a short distance from each other, can be considered far apart based on vertex-based distances. Our algorithm deals successfully with this issue by clustering subtrajectories using a similarity measure (a variant of the Fréchet distance as discussed in Section 2) that treats the trajectories as continuous objects rather than as discrete point sets.

Missing trajectory parts. The use of proximity thresholds has the risk of ignoring whole parts of trajectories when they are too close to others, as in Figure 8e. In our algorithm, the bundles cover all input trajectories in their entirety. The only way for a trajectory not to be fully repre-

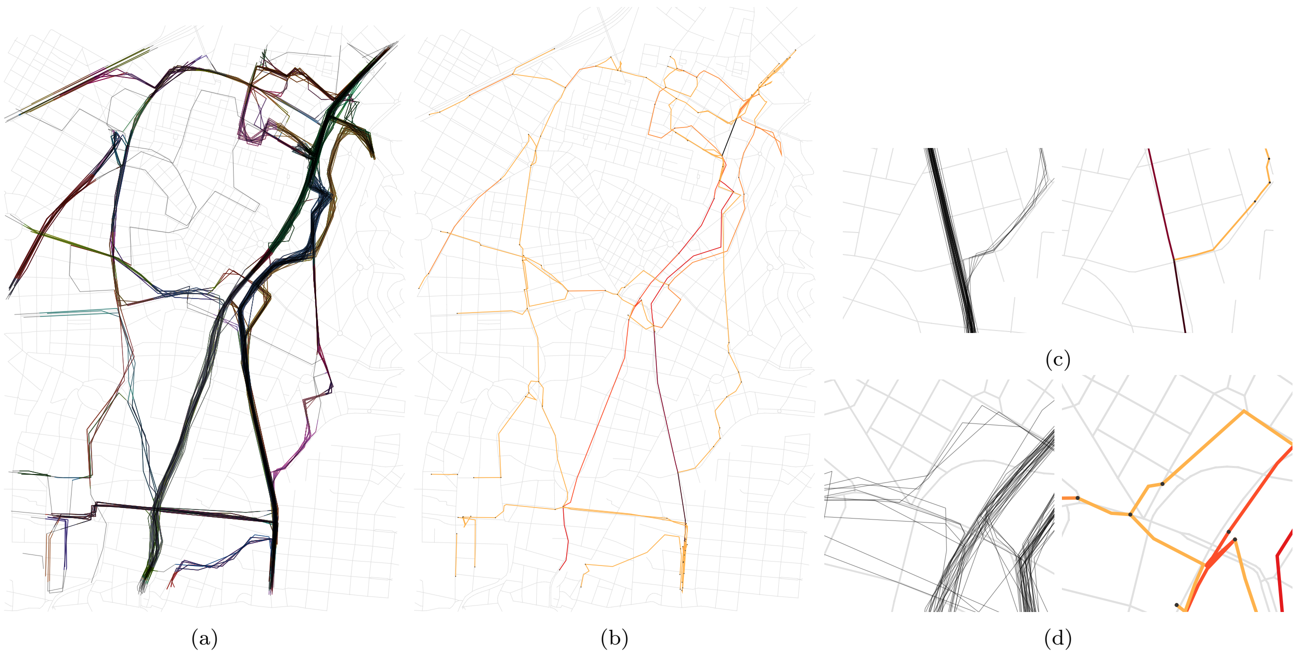

Figure 6: The input trajectories and detected bundles (a), and the constructed map (b) for the Athens-small data set. The corresponding map from OpenStreetMap is displayed in gray. We see that roads with different densities in the trajectories are properly represented in the map (c), and that many intersections are identified correctly, even if the trajectories are spread apart (d).

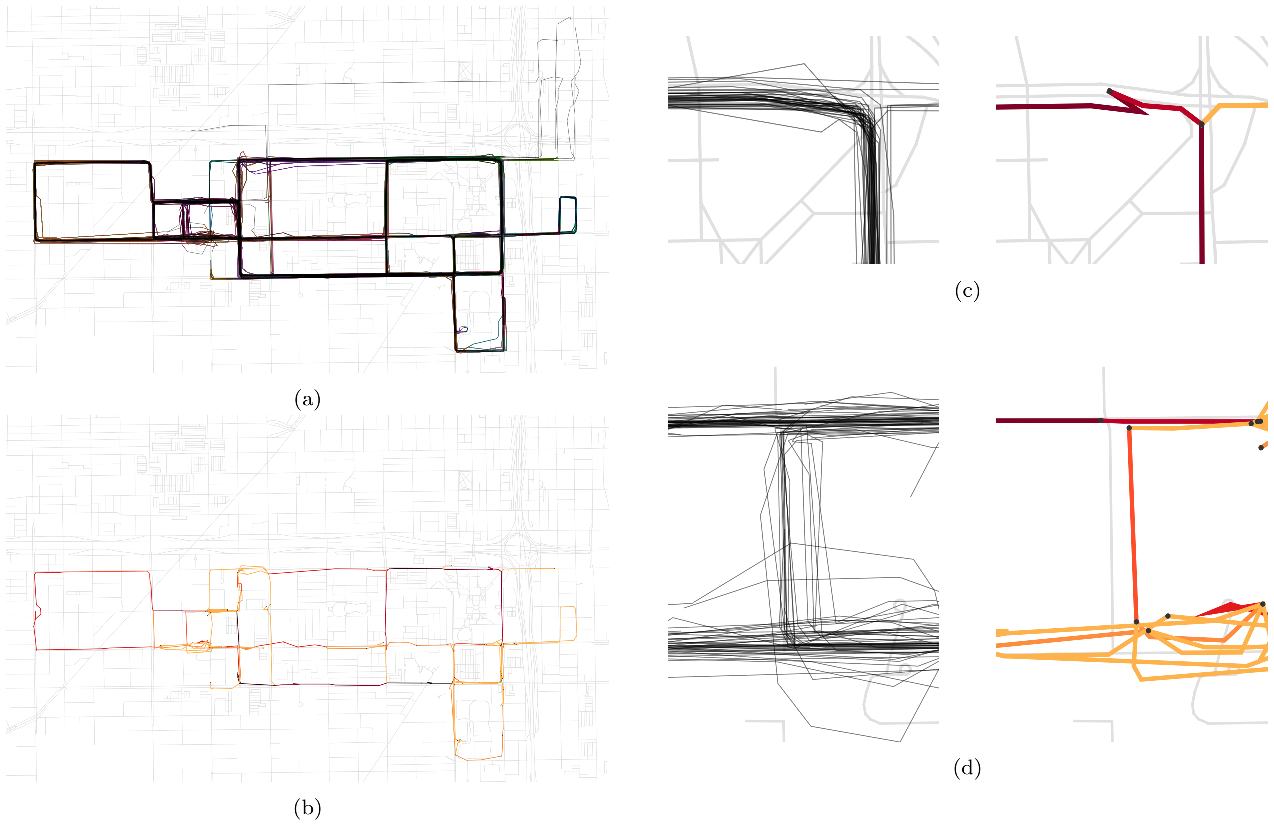

Figure 7: The detected bundles (a), and the constructed map (b) for the Chicago data set. Detail on two situations that are difficult for most existing algorithms: noisy trajectories turning at an intersection (c) and a relatively short edge between two parallel roads combined with noise (d).

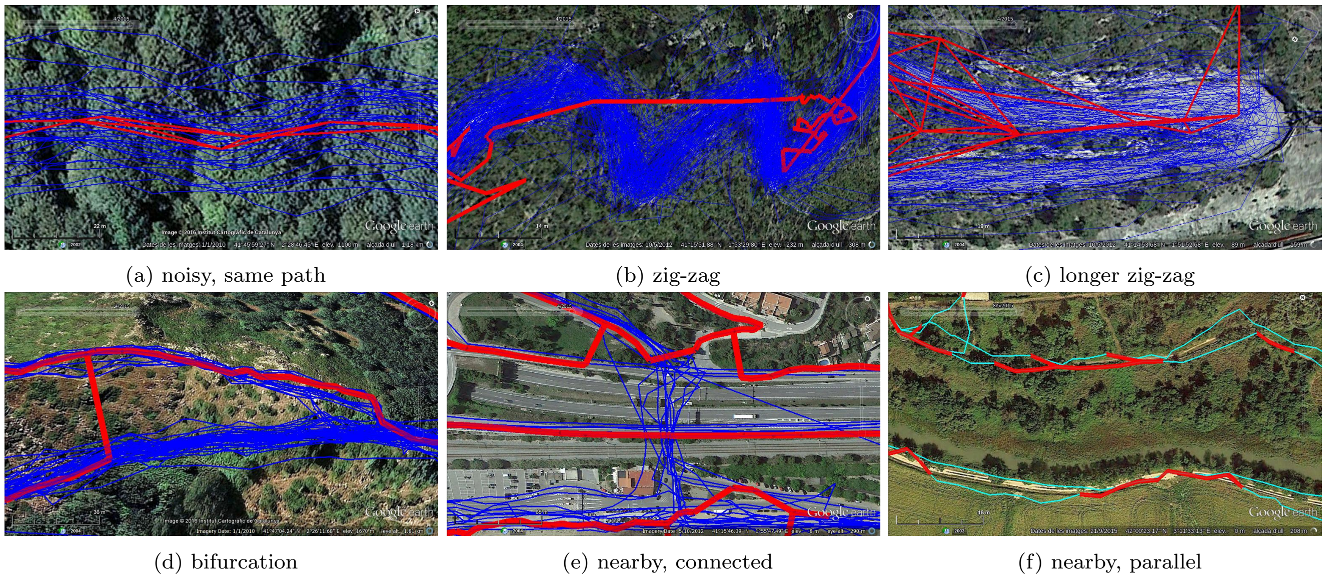
(a) noisy, same path (b) zig-zag (c) longer zig-zag (d) bifurcation (e) nearby, connected (f) nearby, parallel

Figure 8: Challenges for map construction algorithms on hiking data (figures from [17]). The blue paths indicate input trajectories, and the red paths the generated map edges from one of the previous algorithms.

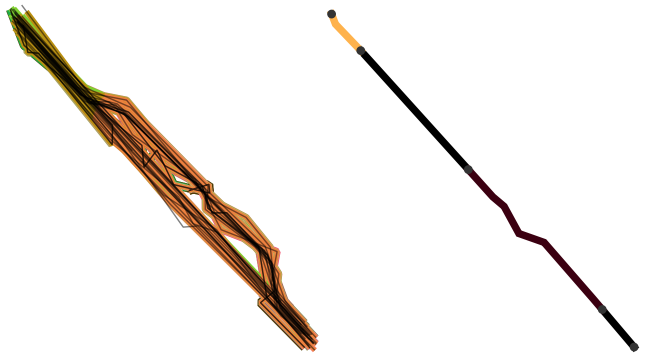

Figure 9: Bundles and map for the trajectories in situation similar to that of Figure 8a. The edges of the map are colored according to the size of the supporting bundle. As the size of the bundles increases, the colors become darker, i.e. small bundles are yellow, large bundles are black.

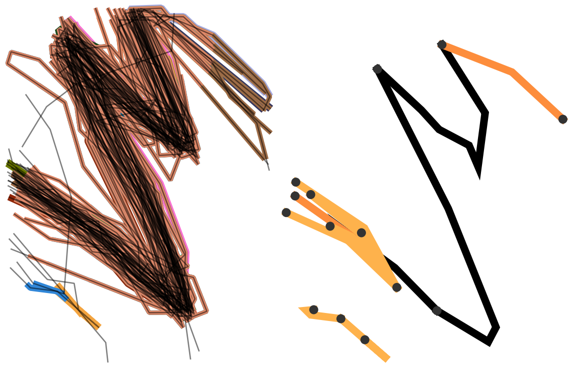

Figure 10: Bundles and map for zigzagging trajectories like in Figure 8b.

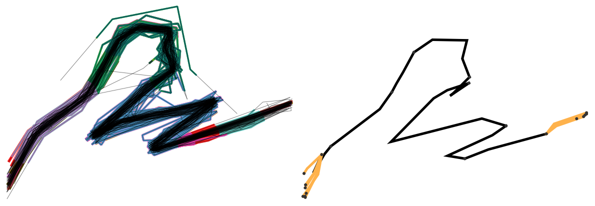

Figure 11: The bundles and map for the trajectories on the high-curvature path whose detail is shown in Figure 8c.

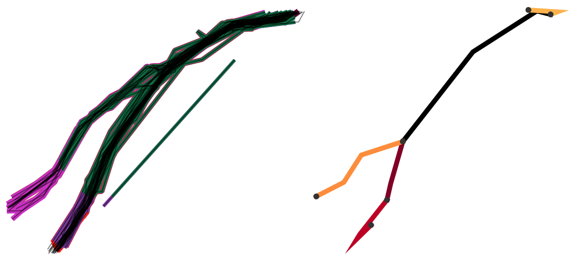

Figure 12: The bundles and map for the trajectories shown in Figure 8d.

sented in the map is to have all bundles containing it considered not significant. This should only occur if there are very few trajectories at that location. Therefore, the situation of Figure 8e is correctly handled by our algorithm (see Figure 13).

Algorithms that require a minimum number of trajectories along each edge of the map have the risk of producing maps with missing edges or fragments. See Figure 8f. In our algorithm, each bundle is represented by a single path. As long as we have a single bundle covering all the relevant trajectories for a given path, we can find get a single, connected path in the constructed map (see Figure 14). It may, however, happen that there are multiple bundles on the same

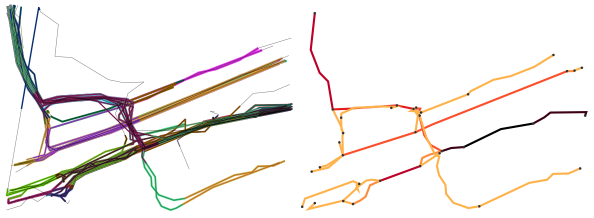

Figure 13: The bundles and map for the trajectories shown in Figure 8e.

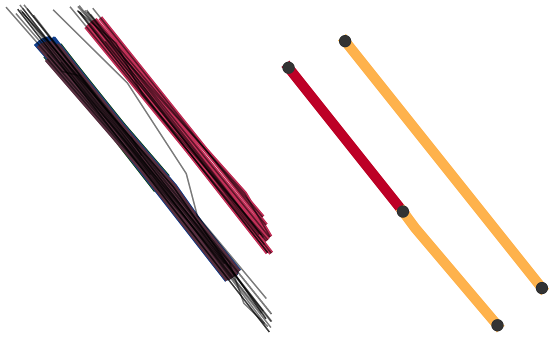

Figure 14: The bundles and map for two sets of parallel trajectories similar to the situation shown in Figure 8f. Our algorithm correctly detects two separate edges.

path, on both sides of some bottleneck on that path, something that could lead to two disconnected edges in the final map. Therefore, the bundle-specific ε must be large enough to cover these (small) bottlenecks.

Figure 15 shows the trajectories, detected bundles, and map generated by our algorithm for a large set of trajectories of the Garraf hiking set. In this data set several of the challenges highlighted in the small samples show up, such as zig-zags, bifurcations, noisy data, and different densities.

3.3 Limitations

The bundling approach of our algorithm is able to deal successfully with many situations that are extremely difficult for previous algorithms. Nevertheless, some situations remain challenging for the current implementation of our algorithm. These challenges are mostly related to how we construct the map from the bundles, rather than from the detection of the bundles themselves.

One of the remaining issues can be seen in Figure 7d, where a lot of noise in the trajectories causes us to detect many small bundles that are all sufficiently different from each other. This causes our algorithm to draw many edges in the map. Our algorithm may also draw many map edges in a small region when the underlying trajectories end, but at quite different positions; see Figure 16 for example. In this case, we cannot find a bundle that bundles the ends of these trajectories together nicely. Instead, we get many smaller bundles grouping trajectories together that end at more or less the same location. Since we have an edge for each bundle, we get many edges instead of a single one. It may be possible to fix such artifacts by a post-processing step in which we would merge edges at the end of different paths when their end-points are close together.

When constructing the map from the bundles (see Section 2.3), we keep track separately of the two locations between which a path should appear, and its representation. After multiple bundles have been processed, the endpoints of the representative may no longer coincide with the desired start and end points. We handle this by connecting the ends of the representative to the desired end points with straight edges. As is visible in Figure 7c this sometimes leads to small visual artifacts at the vertices: we may get small Y-shaped T-crossings and "Z/N-shapes" may appear in places where edges produced by different bundles are joined. A better way for connecting the representatives of the different bundles to each other could probably alleviate this issue.

Since we use a portion of an input trajectory as a way to represent edges or paths on the map geometrically, any noise that appears on that trajectory may also appear in our resulting map. For example, in Figure 17, the blue trajectory that has been chosen as a representative contains a small dent. This dent also shows up in our drawing of the map.

A possible solution for this problem is to choose a different representative. Our current implementation chooses the "cluster center" produced by the bundling algorithm. However, there are various other possible representative trajectories [9, 11]. We ran some initial experiments using the simple median defined by Buchin et al. [9]. While this gives improved results in some situations, the complexity of the median may be high (see Figure 18). This makes the drawing of the map look cluttered. Hence, an other simplification step may be required.

4. CONCLUDING REMARKS

We presented a map construction algorithm based on subtrajectory clustering. The focus of our work was on automatically selecting relevant bundles. As demonstrated in the experimental evaluation, this successfully allows us to address key challenges in map construction. Specifically, by working with subtrajectories rather than points, the algorithm easily handles irregular sampling rates and manages to separate roads that are close in proximity, but separate. By looking for stable clusters, we avoid common problems of map construction algorithms that use a global spatial proximity parameter.

Our algorithm does not explicitly consider outliers, but successfully identifies outlier (sub)trajectories, since these result in bundles of small size, allowing us to easily prune outliers. This pruning currently requires an additional parameter, namely, the minimum size of a relevant bundle. The open problem remains as to how (or if) one could automatically choose this parameter. For instance, small bundles that are fairly close to a large bundle may be considered less relevant than a small bundle with no other bundles nearby.

Currently, we generate many (maximal) candidate bundles, and then test for λ-subbundles. To generate the candidate bundles, we modified the clustering algorithm [8], which has a runtime of roughly $O(n^2)$, where n is the size of the complete data set. On small data sets this works reasonably well, but for larger data sets we need a more efficient algorithm. Hence, one of the main topics for future work would be to design and implement a more efficient algorithm that directly computes the set of maximal stable large bundles that we are interested in.

For hiking data, pedestrians usually use the same path in both directions; whereas, in vehicular data, the range of pos-

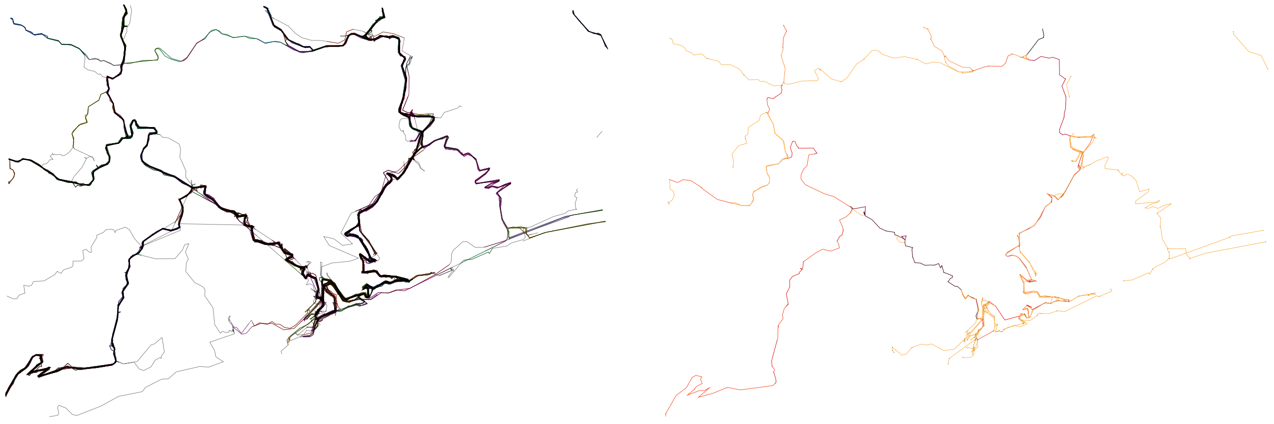

Figure 15: The bundles and map for a subset of 140 trajectories, and a total of 7469 vertices, of the Garraf hiking data set.

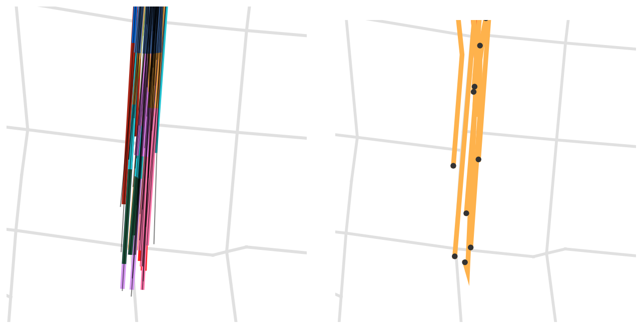

Figure 16: Many trajectories ending on the same road at different positions, giving multiple small bundles instead of a large one (left). This leads to many edges for each bundle (right).

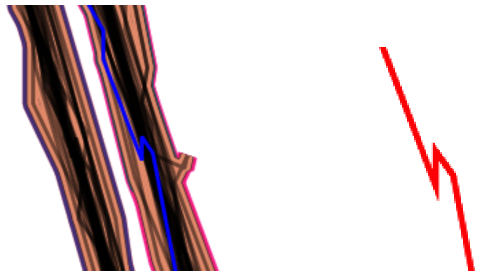

Figure 17: Error in the representative of a bundle is reflected back in the map

sibilities is wider and it is not uncommon to find one-way or bi-directional single- or multi-lane streets. Some of the previous algorithms make strong assumptions (e.g., right-hand traffic with one-way lanes) and produce artifacts when dealing with other settings. Our algorithm is capable of working with the one-way or two-way trajectory directions by just adding the reversed trajectories appropriately. It would be interesting to explore how the direction of movement could be used more explicitly in the setting of hiking data.

Finally, our algorithm stitches together relevant bundles to obtain the map. For this, we use a relatively simple greedy strategy. The portion of the map corresponding to a bundle is represented by one of the subtrajectories in the bundle. In

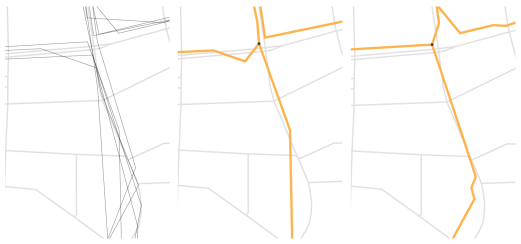

Figure 18: The median (right) in comparison to the representative of the bundles (middle) avoids outliers, but also adds more vertices, in particular making paths less straight. Input trajectories on the left.

most cases, this works well, due, in part, to the fact that the algorithm for subtrajectory clustering naturally computes a representative for each bundle. However, as noted in the previous section, the representative itself may contains noise. Understanding (both theoretically and experimentally) the implications of using different algorithms for choosing a bundle representative is another direction of future research. In the stitching process, we do not explicitly handle road crossings, resulting in small dents at T-crossings. Crossings in principle result in bundles of subtrajectories of large size but small length. Currently, we do not make use of these bundles, but leave this to future research.

Acknowledgments

This research was initiated at a research visit of some of the authors in Barcelona in 2015. The research was continued at the Dagstuhl seminar 16022 "Geometric and Graph-based Approaches to Collective Motion", the McGill-UVic-INRIA Workshop on Geometry and its Applications, and Dagstuhl Seminar 17072 "Applications of Topology to the Analysis of 1-Dimensional Objects". We thank all organizers for providing the environment for these research discussions. K.B. is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.207. V.S. and R.S. are partially supported by projects MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR2014SGR46. R.S. is also supported by MINECO through the Ramón y Cajal program. C.W. is supported by NSF CCF-1637576. F.S. was supported by the Danish National Research Foundation under grant nr. DNRF84.

5. REFERENCES

[1] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. J. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. In *Proc. 27th ACM Symp. on Comp. Geometry*, pages 37–46, 2011.

[2] M. Ahmed, B. T. Fasy, M. Gibson, and C. Wenk. Choosing thresholds for density-based map construction algorithms. In *Proc. 23rd Int. Conf. on Geographic Information Systems*, page 24. ACM, 2015.

[3] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

[4] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. *Map Construction Algorithms*. Springer, 2015.

[5] M. Ahmed and C. Wenk. Constructing street networks from GPS trajectories. In *Proc. 20th Ann. European Symp. on Algorithms*, pages 60–71, 2012.

[6] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1&2):75–91, 1995.

[7] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proc. 20th Int. Conf. on Advances in Geographic Information Systems*, pages 79–88, 2012.

[8] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *Int. J. Comput. Geom. & Appl.*, 21(03):253–282, 2011.

[9] K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

[10] L. Cao and J. Krumm. From GPS traces to a routable road map. In *Proc. 17th Int. Conf. on Advances in Geographic Information Systems*, pages 3–12, 2009.

[11] E. Chambers, I. Kostitsyna, M. Löffler, and F. Staals. Homotopy measures for representative trajectories. In *Proc. 24th European Symp. on Algorithms*, pages 27:1–27:17, 2016.

[12] F. Chazal, R. Huang, and J. Sun. Gromov-Hausdorff approximation of filament structure using Reeb-type graph. *Discrete and Computational Geometry*, 53(3):621–649, 2015.

[13] C. Chen and Y. Cheng. Roads digital map generation with multi-track GPS data. In *Proc. Workshops on Education Technology and Training, and on Geoscience and Remote Sensing*, pages 508–511. IEEE, 2008.

[14] D. Chen, L. J. Guibas, J. E. Hershberger, and J. Sun. Road network reconstruction for organizing paths. In *Proc. 21st ACM-SIAM Symp. on Discrete Algorithms*, pages 1309–1320, 2010.

[15] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, Oct. 2006.

[16] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[17] D. Duran, V. Sacristán, and R. I. Silveira. Map construction algorithms: an evaluation through hiking data. In *Proc. 5th Int. Workshop on Mobile Geographic Information Systems*, pages 74–83, 2016.

[18] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective*, pages 128–151. Springer, 2003.

[19] A. Fathi and J. Krumm. Detecting road intersections from GPS traces. In *Proc. 6th Int. Conf. on Geographic Information Systems*, pages 56–69, 2010.

[20] X. Ge, I. Safa, M. Belkin, and Y. Wang. Data skeletonization via Reeb graphs. In *Proc. 25th Ann. Conf. on Neural Information Processing Systems*, pages 837–845, 2011.

[21] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive GPS traces data. In *Proc. IEEE Int. Geoscience and Remote Sensing Symp.*, pages 667–670, 2007.

[22] R. Jacobs. Constructing Maps by Clustering Trajectories. Master's thesis, TU Eindhoven, 2016.

[23] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proc. 20th Int. Conf. on Advances in Geographic Information Systems*, pages 89–98, 2012.

[24] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2007.

[25] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 669–677, 2012.

[26] B. Niehofer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In *Proc. 1st Int. Conf. on Advances in Satellite and Space Communications*, pages 156–161, 2009.

[27] M. Quddus, W. Ochieng, and R. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, pages 312–328, 2007.

[28] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive GPS vehicle trajectories. In *Proc. 12th Int. IEEE Conf. on Intelligent Transportation Systems*, pages 48–53, 2009.

[29] A. Steiner and A. Leonhardt. Map generation algorithm using low frequency vehicle position data. In *Proc. 90th Ann. Meeting of the Transportation Research Board*, pages 1–17, January 2011.

[30] R. van Leusden. A novel algorithm for computing the Fréchet distance. Master's thesis, TU Eindhoven, 2013.

[31] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proc. 23rd Int. Conf. on Advances in Geographic Information Systems*, page 10 pages, 2015.

[32] S. Worrall and E. Nebot. Automated process for generating digitised maps through GPS data compression. In *Proc. Australasian Conf. on Robotics and Automation*, 2007.