Use of GPS Data and Visual Summaries for Traffic Planning

Bram Custers, Wouter Meulemans, Bettina Speckmann, Kevin Verbeek
Eindhoven University of Technology
The Netherlands

ABSTRACT: *In the road networks, the GPS data provide valuable traffic data for planning. In this process, more volume of data is impactful that provide challenges while observing significant features. The GPS and the road network can be integrated to arrive a viable solution and generating visual summaries. Now we outlined acoordinated fully-automated pipeline for computing a schematic overview of mobility patterns from a collection of trajectories on a street network. The proposed framework used the known building blocks from GIS, automated cartography, and trajectory analysis: map matching, road selection, schematization, movement patterns, and metro-map style rendering. The propositions are experimented by subjecting two cities where the real-world data is used to assess the pattern.*

Keywords: Trajectories, Visualization, Schematization

Received: 26 June 2021, Revised 4 September 2021, Accepted 29 September 2021

DOI: 10.6025/jnt/2021/12/4/108-122

1. Introduction

GPS tracks from moving vehicles are an important source of information for traffic analysis and urban planning. Their general ubiquity allows decision makers to understand the usage of transportation networks and urban spaces. However, the sheer volume of the data makes it challenging to render trajectory collections in a meaningful way as to show the general, overarching patterns. Simply plotting all trajectories results in the infamous "spaghetti heaps". Heat maps [15, 19] and other aggregation techniques such as Voronoi aggregation [1] are helpful to "untangle" traffic locally, but they generally fail to capture structural patterns, such as important longer routes.

Summarizing trajectory collections visually, such that salient patterns emerge, inherently requires a form of aggregation or simplification of the data. That is, the level of detail and information shown should be scale-appropriate and avoid a cognitive overload, while still being able to provide insight into the overall mobility. There are various techniques to cluster trajectories and compute a representative for visualization [6, 16], or to simplify trajectories [12]. However, these techniques typically focus on trajectories in a general 2D space, whereas our focus lies on trajectory data on transportation networks, specifically vehicles on a road network. As such, the problem changes in nature, as selecting representative routes and performing simplification needs to be "network-aware". Stronger still, to reduce the visual complexity of the eventual visu-

alization, not only the trajectories need to be simplified, but also the underlying street network. To arrive at meaningful results which show traffic patterns in the correct context, the simplification and aggregation of the trajectory collection and of the network have to go hand-in-hand: they need to be *coordinated*.

Contribution and organization. We propose a coordinated fully-automated pipeline for computing a schematic overview of mobility patterns. Our pipeline consists of five steps, each utilizing well-known building blocks from GIS, automated cartography and trajectory analysis: map matching, road selection, schematization, movement patterns, and metro-map style rendering. We present our overall pipeline and its rationale in Section 2; the subsequent sections describe each step in more detail. Each of these sections also describes how we implemented the corresponding step in our proof-of-concept. For illustration we use a realworld dataset of vehicle trajectories around The Hague in the Netherlands. In Section 8 we discuss the results of our pipeline using a second real-world data set around Beijing. We close with a general discussion of our pipeline and future work in Section 9.

Related Work. We focus here on related work pertaining to the visualization of large volumes of trajectories and discuss related work for each step of the pipeline in the respective section. Most research in this area aims to provide an overview of space usage, without showing or using the temporal component of trajectories; see the two extensive surveys by Chen et al. [10] and Andrienko et al. [3]. The notable exception are space-time cubes [13, 22], though they do not scale well to large numbers of trajectories without some form of aggregation.

To identify larger patterns, one can focus on visualizing the origin-destination data only, that is, focus only on the endpoints of the trajectories, possibly with some form of spatial aggregation. There are various techniques to visualize such information, e.g., [24, 31, 34]. Visualizing OD-data shows patterns beyond local traffic, but typically does not show any information on the actual routes. As such it does not support understanding mobility from the viewpoint of traveling through a network. Indeed, these techniques are typically applied in situations where the exact trajectories or routes are not available or not of interest.

2. The Pipeline

Our input is a set T of trajectories, and a network G. Our goal is a schematized representation of G together with the most salient mobility patterns in T. Before we can describe our pipeline in more detail, we first give the necessary definitions.

Definitions. A *trajectory* is a sequence of measurements $T = \langle (x_1, y_1, t_1), \ldots, (x_n, y_n, t_k) \rangle$ with a position $(x_i, y_i) \in \mathbb{R}^2$ at each timestamp $t_i \in \mathbb{R}$. We restrict our attention to the spatial domain and hence we ignore the temporal component beyond providing an ordering of the measurements. Thus, for the purpose of this paper, a trajectory T is a (directed) polygonal line with vertices (x_i, y_i).

A *(road) network* is a directed graph $G = (V, E)$ where each vertex has a location in $\mathbb{R}^2$. Edges have an associated geometry connected to their endpoints; initially, this is typically the unique line segment, but during our pipeline, this geometry can change. We further assume that each road has an associated road type, which is ordinal (e.g., "highway").

We assume that the network is plane, that is, the edges do not cross except at common endpoints. This condition is not generally satisfied by actual road networks, but we can introduce extra vertices on these intersections to planarize the network. By marking these extra vertices, we can perform algorithms both on the planarized and original network. We further use $|G|$ to denote the complexity of the network, measured as its number of edges, though note that this assumption implies that $|G| = |E| = O(|V|)$.

We use *route* to refer to a (directed) path in the network. For example, map-matching a trajectory results in its route: the path in the network that was traversed by the vehicle captured by the (noisy) trajectory. We refer to such a path as *the* route of the trajectory.

A route does not have to correspond to a single or entire trajectory. Specifically, we say that a route is *supported by* a trajectory T if it is a subpath of the route of T. We use *bundle* to indicate a route that is supported by multiple trajectories,

using the *support of a bundle* to indicate the number of supporting trajectories. Bundles should aim to capture mobility patterns; precise criteria to form meaningful bundles are discussed in Section 6.

The pipeline. Our goal is to compute a schematic representation of G with salient bundles that are "supported by" T. To achieve this, our pipeline consists of five steps, briefly sketched below. See Figure 1 for an example of the results of these steps. In the subsequent sections we discuss each in more detail. Important in our treatment of the network and the trajectory information is to coordinate changes: specifically, changes in the network should be translated to changes in the trajectory information.

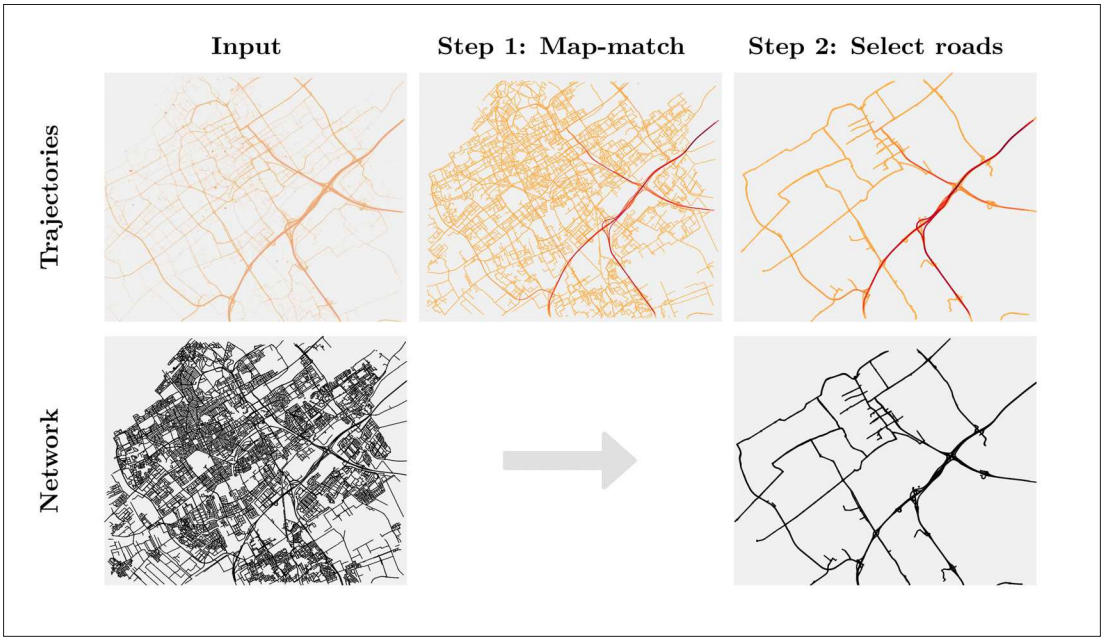
Step 1: Map-match We aim to visualize mobility patterns via bundles, frequent routes in the data. However, trajectories are not the same as routes, and thus cannot support a bundle. As such, we first map-match the trajectories to the network. The minimal input to this step is a single trajectory and the network, such that each trajectory can be processed individually. Map-matching computes the route associated with this trajectory. The result of this step is a set R of routes, rather than trajectories. In our implementation, the trajectories are not used further. This step enables our pipeline to coordinate changes in the network and the routes.

Step 2: Select roads A typical road network is very detailed, much beyond the level of detail that we need to visualize the general mobility patterns and well-supported bundles. The minimal input to this step is the road network and the set of routes derived in Step 1. Note that we could, in principle, base selection purely on the network, and adapt the routes as necessary. However, we also choose to include all parts of the network which are frequented heavily by the routes. The result is a subset of the network and a mapping of the original routes to this selected network.

Step 3: Schematize To reinforce the summarizing nature of the eventual visualization, we reduce the visual complexity of the selected network via schematization. The input is the selected network and the mapping of the routes. The output is a strongly simplified version of this network. The mapping of the routes is maintained (coordinated) during this process. Optionally, the edges of this schematic network may be annotated with information about the length of the edge for the purpose of bundling.

Step 4: Bundle In our schematic representation, we find well-supported bundles. The input is thus the schematic network and the mapping of routes to this schematic representation. The output is a set of bundles that are well supported.

Step 5: Render We now have all ingredients for our visualization: the schematic network as well as salient mobility patterns (bundles). The result is the eventual visualization which shows these two pieces of information effectively.



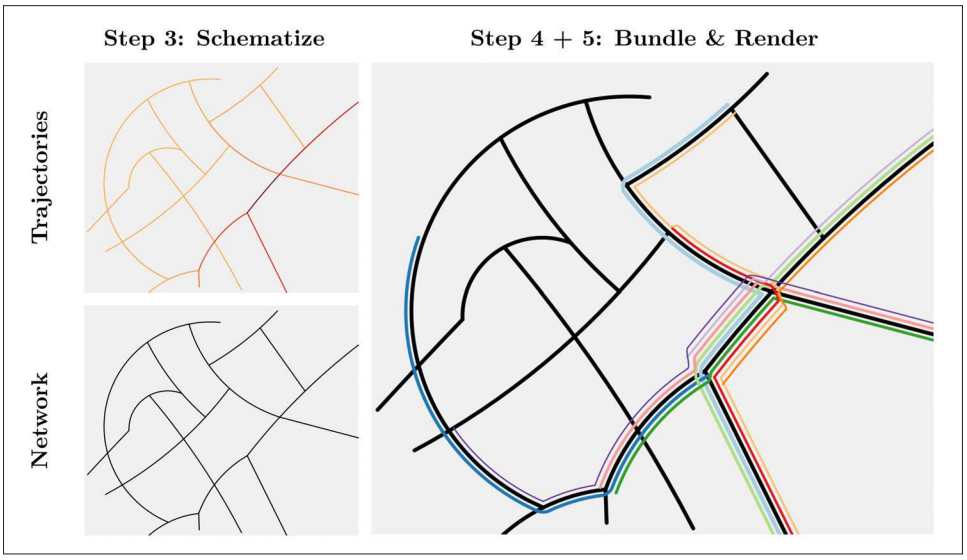


Figure 1. Our pipeline for coordinated schematization on the Hague dataset. The input trajectories are shown as a density map. For the map-matched routes, we use a orange to red scale to convey low to high traffic volume per edge. We compute the bundles in Step 4 with $S_{min} = 500$, $L_{min} = 10000m$, $p = 0.5$ and shrunk edge lengths, see Section 6 for more details on these parameters

3. Step 1: Map-match Trajectories to the Network

Desiderata. To eventually visualize common routes in the network, we must ensure that our trajectory information is mapped to the network, that is, that each trajectory is translated into a route. This problem is generally known as *map matching*.

Related Work. Map-matching is a broadly studied topic in GIS; see [8] for a recent survey. As discussed there, the different approaches are categorized by their matching model. For our purpose, we consider a map-matching approach using the similarity model, since this requires less parameters. But in principle, any approach will work in our pipeline.

Our implementation. We use the map-matching algorithm described by Alt et al. [2], which runs in $O(|G||T| \log |G||T| \log |G|)$ time on a network G and trajectory T. The algorithm ensures that the route found has minimal Fréchet distance to the trajectory and thus that it is geometrically close. Our motivation for doing so is to remain as close as possible to the information of the trajectory. That is, to the information "visible" if we were to simply draw all trajectories. This algorithm is able to handle noise relatively well, but in case of very sparsely sampled trajectory data, it may struggle to find the most natural route.

4. Step 2: Select roads

Desiderata. We aim to select the roads for two somewhat distinct purposes. First, we want to select the roads where there is considerable traffic, to facilitate well-supported bundles. This purpose is thus inherently data-driven. But second, we want to select major roads to provide a frame of reference for the viewer as to how the mobility patterns are situated in space. It stands to reason that often, major roads also carry a large part of the traffic. However, this is not necessarily the case.

Related work. Selection is an important part of road network generalization algorithms. The goal is to select the most important parts of the network, such that the remainder can be discarded in the simplification process. Different approaches exist to determine what features of the road-network are "salient". Examples of these approaches are using the mesh density [9], using user defined weights [14] and using areas of faces combined with semantic labels [20]. Different from the previous are approaches that focus on "strokes" through the network: lines of good continuation, that is, lines with small local curvature [21]. During generalization, these strokes are considered atomic units and are selected based on their relative importance. This importance is often determined via network centrality measures [29, 33].

More recently, approaches focus on using traffic data to inform the selection process [23, 35]. Yu et al. propose an approach that is based on strokes, but during the selection process considers traffic flow from one stroke to the other, increasing the likelihood that strokes that give good traffic flow continuation are selected together. Van de Kerkhof et al. [23] follow a different approach, where the selection process is formulated as a covering problem, and the trajectories need to be covered by the selection of the road-network.

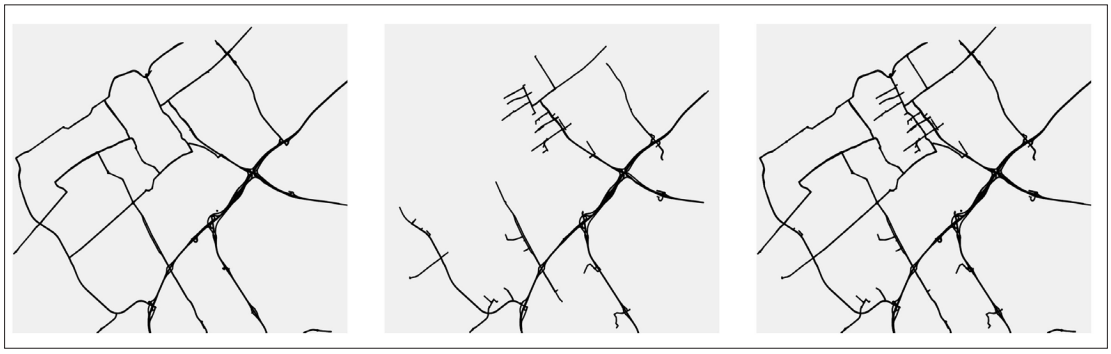


Figure 2. Selection by road type (left) and by traffic (middle), combination of both (right)

Our implementation. We use the approach by van de Kerkhof et al. [23] which seeks to select a subgraph $G2$ of the network with bounded length, such that the number of routes that are completely within this subgraph is maximized. Though the authors prove that this problem is NP-hard, they also describe a heuristic that runs in $O(|\mathrm{T}|2 \log |\mathrm{T}| + |\mathrm{T}||G|)$ time; we use this heuristic in our implementation (see Fig. 2 (middle)). After selecting $G2$, we add any edges that were not selected yet and have a large enough road type. If the resulting selected network is not connected, we optionally select the largest connected component; none of the later steps require the network to be connected (see Fig. 2 (left) and (right)).

5. Step 3: Schematize the Network

Desiderata. Even after selection, the network tends to contain more detail than necessary to provide a meaningful overview of mobility patterns: different lanes, cloverleaves, etc. Instead, we should get a high-level overview that communicates the main connectivity in the network, and as such create space to visualize mobility patterns (bundles). That is, we should collapse (aggregate) and simplify such local details. We do so beyond the need of target scale, instead focusing on *functional* detail: that is, we schematize the network. The network should remain spatially informative: roughly similar to the overall input geometry.

Related work. In automated cartography, the process of schematization is used to render aesthetically pleasing networks or polygonal domains that are decluttered enough to convey important information on the schematic [17]. Compared to generalization, schematization commonly reduces the input to such an extent that it is not necessarily realistic anymore, albeit retaining important features to recognize the original. Note that in general, the input to these algorithms is a detailed map of the road-network, whereas we input a selection of the map based on data, thus making it a data-driven schematization.

One approach is to limit the type of the geometry in the output, thus naturally reducing detail. Here it is common to fix the allowed number of orientations of lines in the network [7, 30], particularly in the context of metro maps. Alternatively one can fix the geometric primitives that can be used, for instance circular arcs [25] or Bézier curves [26].

To retain recognizability, schematization typically limits the spatial distortion between input and output by fixing vertex locations [7] or minimizing the distance between input and output edges, for instance via the Fréchet distance [25]. In addition, it is common to maintain the topology of the input, which plays a key role in recognition of the output. We note that for our approach, we want to retain the topology of the network at a certain scale, thus small topological features should be removed prior to applying a topology-preserving schematization approach. An alternative would be to consider continuous scale generalizations [27, 28] and applying schematization at the desired scale. An important aspect is then to be able to retain a mapping from the edges in the selected network to the schematization.

Our implementation. We first drastically simplify and collapse the selected network $G2$, after which we apply the arc-

schematization algorithm by van Dijk et al. [25] for its aesthetic and clean representation, resulting in a schematic road-network $\mathcal{G}$.

Our implementation applies the sequence of operations described below. We use simple steps in an incremental fashion to facilitate coordination and maintain a mapping between the edges of the selected network and the schematic network. Our simple operations can result in fairly coarse approximations. However, since our target is a highly abstracted final map, the coarseness of the earlier operations is not an issue.

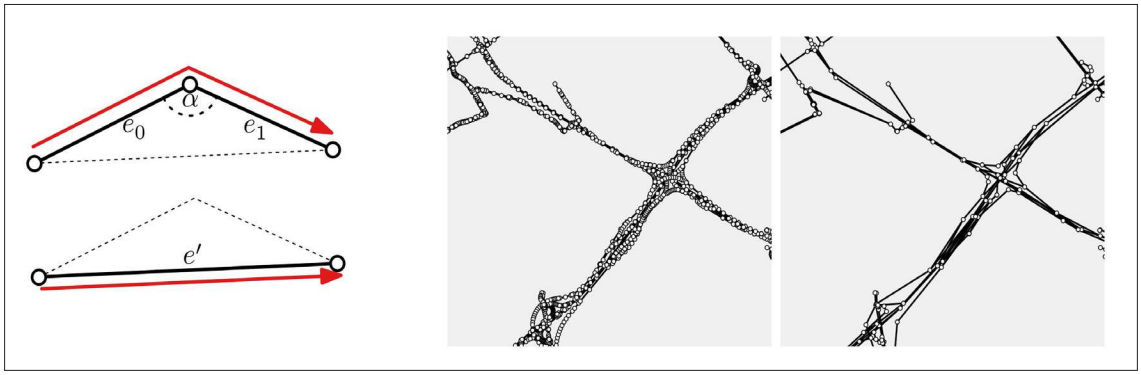


Figure 3. (left) Replacing shallow turns; (middle) network before; (right) network after

Collapse dead ends We first remove short paths in the network that do not increase the overall connectivity. Starting at a degree-1 vertex, we move along degree-2 vertices only to trace a visual "dead end" until we find a vertex that does not have degree 2. We compare its geometric length to a predefined parameter l_{max}, and collapse the path to this last vertex if its length falls below this threshold. We use $l_{max} = 100m$ initially, and $l_{max} = 1000m$ after the face-collapse operation. For coordination, we reroute any route along the collapsed edges to the endpoint that remains.

Replace shallow turns A detailed input network frequently contains small bends. The final schematization would remove such detail, but we perform this step early to simplify the merge and collapse operations to follow. For every degree-2 vertex, we consider the smallest angle between its incident edges. If this angle exceeds some predefined limit β, we replace the vertex and its two incident edges with a single edge; see Figure 3. We use $\beta = 150^o$ before and $\beta = 140^o$ after the face-collapse step. For coordination, we reroute any route on one or both of the replaced edges e_0 and e_1 to the new edge e'.

Merging vertices Junctions in the road-network are too detailed for our schematic; ideally we represent them by a single vertex. To this end, we fix a radius r within which we merge vertices. For a vertex v, the merge operation for v merges all vertices within distance r of v (including v itself) to a single new vertex, placed at the centroid of the merged vertices (see Figure 4). We iteratively merge vertices, prioritized by the number of vertices within radius r. We use $r = 0.01D$ with D the length of the diagonal of the bounding box of the network. After merging faces, we use a larger radius of $r = 0.03D$.

For coordination, edges inside the merge radius are mapped to the new vertex (e.g., red edges to v_r in figure). Edges between different new vertices or between a new vertex and an unmerged vertex are consolidated to new edges (purple and blue in figure).

Merging vertices with edges A vertex can be close to an edge, even though it is not close to its endpoints. In such cases, we merge vertices into nearby edges that are not incident to the vertex itself. We use the distance r as specified earlier, and try to collapse a vertex v to the closest non-incident edge e that is within distance r, where we demand that v lies in the slab spanned by e (see Figure 5). For coordination, any route using the former edge e is now using the two (possibly new) edges e'_0 and e'_1 from v to the endpoints of e.

Face collapse We now consider the faces of the network and collapse them onto a single geometry, if they are "small". Specifically, we collapse faces with an area of at most $a_{max} = 0.01A$, where A is the area of the bounding box of the network.

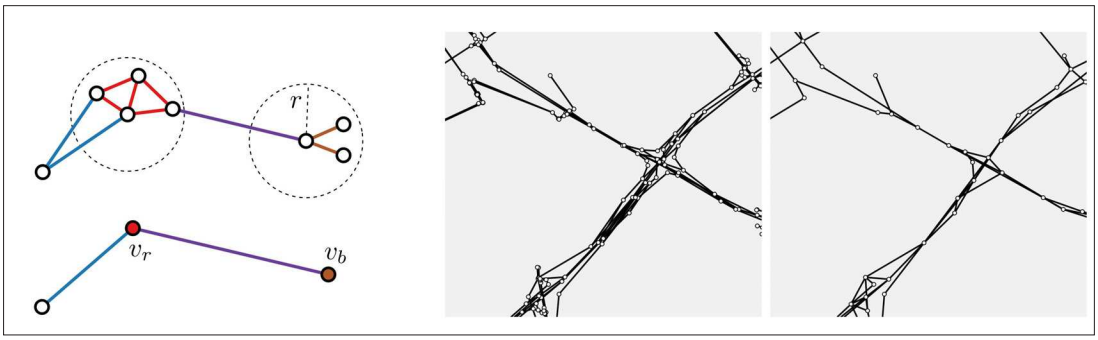


Figure 4. (left) Merging vertices; (middle) network before; (right) network after

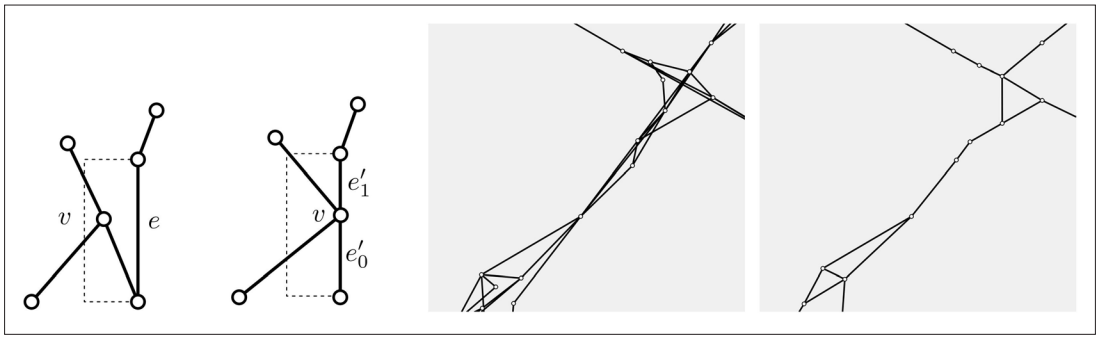


Figure 5. (left) vertex-edge merging; (middle) network before; (right) network after

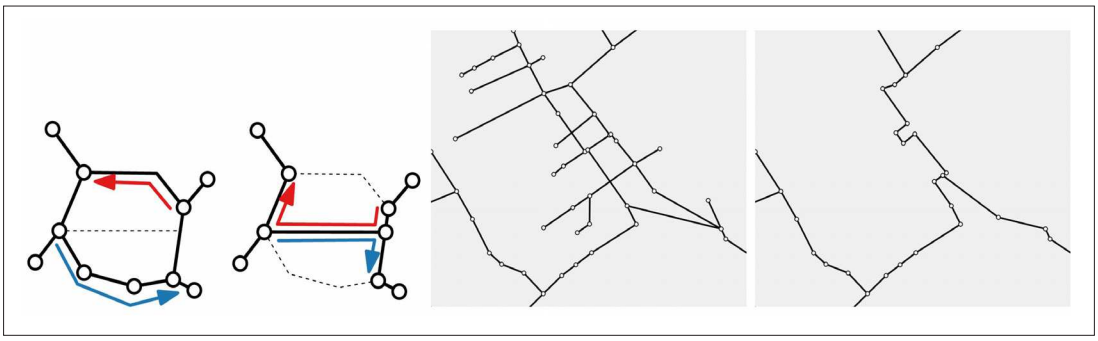


Figure 6. (left) Face collapse; (middle) network before; (right) network after

To collapse a small face F, we first collapse all interior degree-2 paths, independent of length. We then compute the minimal oriented bounding box of F, select its major axis, and cut F along this axis. In case of a non-convex face, we select the longest internal intersection between F and the axis (see Fig. 6). The cut introduces two cycles; we break both cycles by removing an edge or a path of degree-2 vertices from each; we can do so while maintaining the distances to the cut along the cycle for all higher-degree vertices. For coordination, we reroute any route that used the removed edges via the cut.

Cleanup After face merging, we repeat all earlier operations once more to clean up the geometry and prepare for computing the final schematization.

Arc schematization For our final schematization, we chose the arc schematization algorithm by van Dijk et al. [25]. This algorithm produces a low complexity schematization using (circular) arcs, while maintaining the topology of network. To encourage the use of straight lines over very shallow arcs, we increase the relative importance of straight lines by reducing the Fréchet distance for straight lines by a factor 0.3. Because this algorithm only removes vertices, coordination is straightforward, similar to replacing shallow turns.

Time Complexity. The simplification operators run in roughly quadratic time with a straightforward implementation. The arc schematization algorithm dominates the operations to simplify the network. Thus, the time complexity is $O(n^2 h \log n)$, where h is the number of vertices with degree higher than three in the input simplification and n the number of vertices, which is greatly reduced from the input at this point. Coordinating the changes between the road network and a trajectory T takes roughly $O(|G||T|)$.

6. Step 4: Detect Bundles

Desiderata. We aim to detect bundles that are well-supported by the routes to provide insight into the overall mobility patterns. As such, we identify two desired properties of a bundle. First, it has to be supported by many routes. Specifically, we do not want a bundle representing a single trajectory (potentially an outlier), but rather the common behavior. Second, a bundle should be long in terms of (geometric) length. We aim to show mobility patterns that describe how vehicles move through the space. A long bundle is more descriptive of behavior than a short one; in the extreme case, a short bundle (even or especially if it is supported by many routes) may consist of one single edge (road segment), which does not help to communicate patterns beyond the existence of considerable local traffic. That is, we want to find long bundles that are supported by many routes.

As we aim to visualize not one but multiple bundles simultaneously, we further use three criteria to assess a set of bundles. First, we restrict our attention to *maximal* bundles only. That is, we consider only bundles to which we cannot add another route for its support, but specifically also not increase its length without having to reduce its support. Second, we generally want to see patterns of mobility that are *spatially diverse*: we prefer having bundles through different parts of the network, if the trajectories allow them. That is, we would like to avoid overlap between bundles as overlap communicates the same (local) behavior.

Third, a bundle fully contained within another may not provide much extra insight into mobility beyond showing a higher support for the contained route. We call a set of bundles *containment-free* if no bundle is a subroute of another. Note that containment-free bundles are not necessarily overlap-free and thus this is different from spatial diversity. Overlap-free bundles are containment-free, but we do not interpret spatial diversity as strict overlap-free.

In a containment-free set of maximal bundles each route may still support multiple bundles. If these are disjoint bundles, then we accept this as a bundle. However, as we are to eventually visualize the bundles, it may be misleading if well-supported bundles that share an edge are only well-supported because they share many trajectories. Thus, we choose to count the support for bundles in a disjoint manner: that is, routes through the overlap of bundles can be counted to support only one of these. We refer to this as the *disjoint* support.

Finally, note that we consider the network bidirectional, in the sense that each edge is present twice, once for each direction of travel. The considerations above should consider the direction of travel. That is, a bundle is directed and can, for example, only be supported by routes in the same direction. Two bundles that use the same set of edges, but travel in opposite direction, would hence be considered overlap-free.

Related Work. Our bundling is closely related to finding groups of trajectories and finding representatives of trajectories, so-called *centers*. In the context of spatio-temporal data, grouping structures are used to find common patterns of mobility [5]. The groups that are constructed essentially are trajectories that move close together for a sufficient amount of time and with enough members in the group. Since we work with map-matched trajectories, we consider routes close when they share edges in the network, and we require bundles to have at least some minimum length.

Note that finding a group does not directly result in finding a representation for the behavior of the group. To represent the behavior of groups, a common approach is to cluster trajectories according to some metric, resulting in centers describing common behaviour for the trajectories that are close to these centers under the used metric. This has been applied to the spatial component under different metrics and algorithmic approaches [6, 16, 18]; see [36] for a more comprehensive overview.

Since we work with map-matched routes, finding the bundles is similar to finding common substrings over a finite alphabet,

where the alphabet is the set of edges and the routes form strings over this alphabet. Finding k-common substrings [4] is particularly related, since it demands a minimum number of matches k, similar to our high support requirement.

Our implementation. We search for a spatially diverse set of up to k bundles, where each bundle is a maximal bundle with a minimum length L_{min} and minimum disjoint support S_{min}. We construct this set by incrementally adding the "most infor mative" bundle.

To quantify this, we define the *importance* of a bundle in a way that allows a trade-off between length, disjoint support and spatial diversity. Specifically, the importance $I(B)$ of a bundle B is defined as $I(B) = L^p|S|^{1-p}$ for $p \in [0, 1]$, where L is the length of the bundle and S is the (disjoint) set of supporting routes. The rationale behind this is that, in the case of $p = 0$ or $p = 1$ we simply prioritize by total support or bundle length, respectively. However, if we set $p = 0.5$, then we prioritize the bundles by the total length of the route-set in its support. This allows a trade-off between length and disjoint support.

We define the length of a bundle B as $\sum_{e \in B} \ell(e)$ for some function l. With $l(e) = |e|$ (the Euclidean length) we promote long bundles directly. As we use disjoint support, there is already some preference for spatially diverse bundles, but we observe that this still leads to very similar bundles. To promote spatial diversity further, we may alter l. Specifically, we also allow for using $l(e) = |e|/(1 + b(e))$ where $b(e)$ is the number of selected bundles already using edge e. That is, conceptually, we "shrink" edges that have been used by other bundles. We thus refer to this setting as *shrunk edge lengths*.

We compute the most important bundle B by using a simple backtracking procedure. We then add B to our bundle set and remove its support from the complete set of routes. We repeat this process until k bundles have been found. If no bundle of sufficient length and support is found but we do not have k bundles yet, we have S_{min} and repeat. We have S_{min} at most twice, creating three *classes* of bundles ("thick", "medium" and "thin" bundles).

Intuitively, focusing on bundle length (higher values of p) is suitable for finding longer bundles that may have less support; useful to investigate longer mobility patterns. Reducing p focuses more on support and thus on patterns that are very frequent. In Figure 7 (rendered according to our final step) we vary p and observe that reducing p (i.e., increasing importance of support) leads to less spatially diverse routes, but increases the bundle classes.

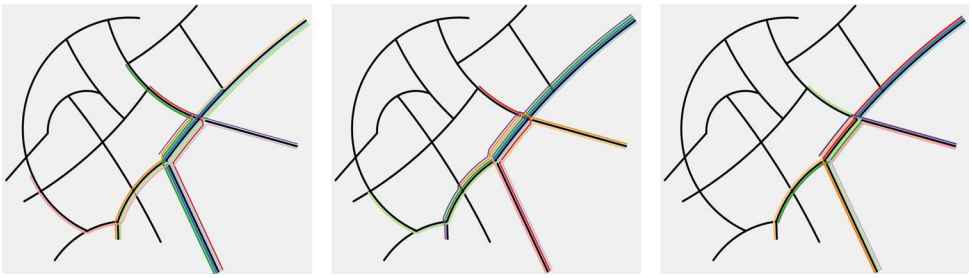


Figure 7. Ten bundles for different importance schemes: $p = 1$ (left), $p = 0.5$ (middle) and $p = 0$ (right). We used $S_{min} = 500$ and $L_{min} = 6000m$, with shrunk edge lengths

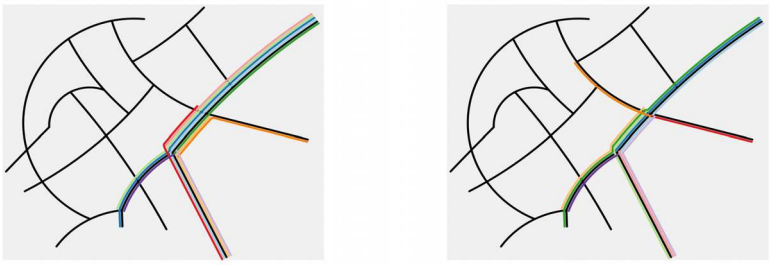


Figure 8. Ten bundles for different approaches to spatial diversity: Euclidean edge lengths (left), shrunk edge lengths (right). We used $S_{min} = 500$ and $L_{min} = 6000m$ and $p = 0.5$

Changing the edge lengths may encourage diversity, but may result in bundles of lower classes, as routes through existing bundles are "shorter" and thus less important. In Fig. 8 we compare the result using both our settings. Euclidean edge lengths show mostly similar routes, but we observe that shrunk edge lengths increase the spatial diversity. It does not avoid overlap, but it does reduce containment slightly (from 5 to 2 contained pairs).

Computing one bundle takes $O(|T||\mathcal{G}|)$ time, where $\mathcal{G}$ is the schematized network. This time includes the necessary changes to the information for computing the next bundle, so computing k bundles takes $O(k|T||\mathcal{G}|)$ time.

7. Step 5: Render schematic network with bundles

Desiderata. We now aim to jointly render the bundles and the schematic network, such that the bundles are clearly conveyed. For this, the bundles should be easily identified. Common practice is to use colors to identify the separate bundles, and in addition separate them visually. However, the bundles need to be rendered such that it is also easy to determine from what parts of the network they originate. Thus, rendering them in close spatial proximity to the associated edges would also be beneficial to the readability of the schematic. In addition, it should be possible to identify the (approximate) support for a bundle.

Related work. Rendering bundles in a network is similar to rendering the lines of a metro network, which is a well studied topic [32]. Common problems involve ordering lines on a common connection to avoid crossings and ensure good continuation at stations.

The offset rendering used in our implementation may want to avoid offsetting edges of a bundle with different distances, thus encouraging good continuation. If we allow gaps between bundles at an edge but disallow differences in offset within a single bundle, the problem is essentially to assign a layer to each bundle, minimizing the number of layers, such that two overlapping bundles use different layers. Even if the network would be a single cycle, this problem is NP-hard [11]; minimizing change is hence NP-hard as well.

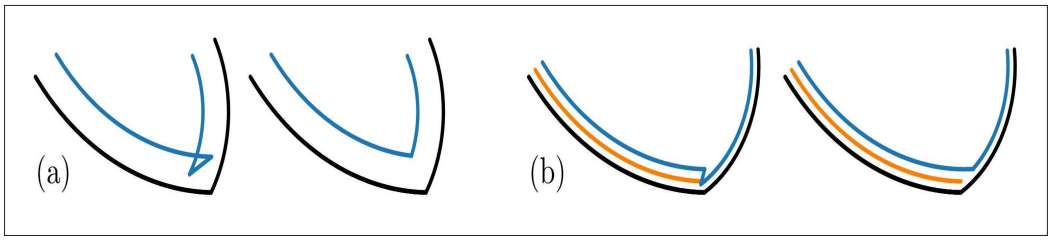


Figure 9. (a) A local self-intersection caused by offsetting and its resolution. (b) Local deviation from the direction of travel of the bundle and its resolution

Our implementation. Our main approach is to render each bundle as a a curve that is slightly offset from the network, such that they do not coincide with the network, nor each other. We visualize the direction of a bundle by offsetting its curves to a particular side of the network edges. As our example datasets are both in areas where cars drive on the right side of the road, we hence locally offset to the right. This scheme allows for identifying the direction of a bundle, without relying on other visual cues such as arrow heads.

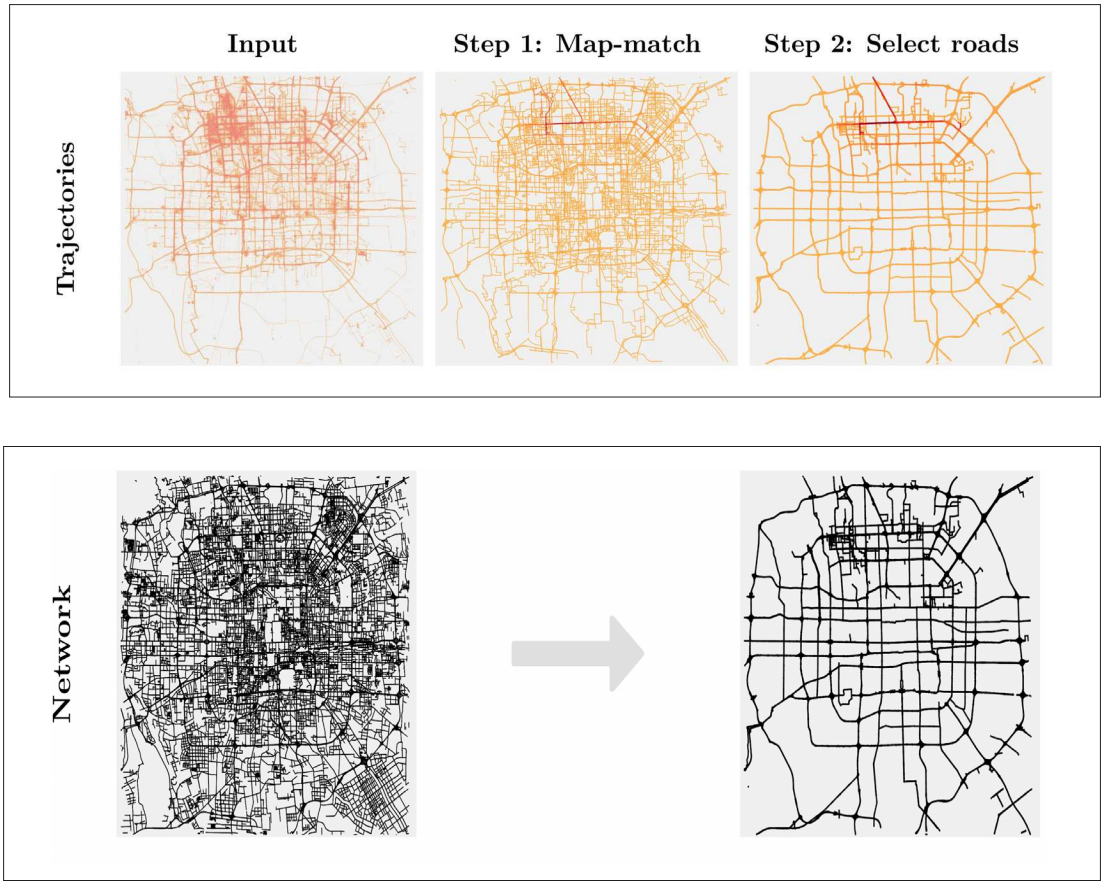
We lay out the bundles one at the time along the network. Each edge in the bundle gives an arc that is offset from the edge by a distance $d(b + 1)$ for some constant $d > 0$, where b is the number bundles already placed at that edge. The bundle is now a sequence of arcs that do not quite connect correctly yet. We initially reconnect the arcs using straight segments. If this causes the curve to locally self-intersect (Figure 9(a)) or cause small corners (Figure 9(b)), directed opposite to the actual bundle direction, we simplify these artifacts as to achieve a simple curve that is always (roughly) directed in the direction of travel of the bundle. This operation takes $O(l \log l)$ time, where l is the total complexity of the rendered bundles. Finally, we slightly smooth the connecting segments by reducing the arcs by a small distance and using the old endpoints as control points for a Bézier curve instead. We render the resulting curves using colors of the "10-class Paired" qualitative scheme from ColorBrewer (https://colorbrewer2.org/), and use a line thickness based on bundle class. By using classes instead of support, it is primarily aimed at separating main from secondary patterns.

8. Results

We implemented our proposed pipeline in C++, using MoveTK (https://movetk.win.tue. nl/) for trajectory processing and CGAL (https://cgal.org/) for geometric operations.

The HR dataset has been used to illustrate and discuss our pipeline throughout the paper. It covers the area of The Hague, the Netherlands. The network is obtained from OpenStreetMap (https://www.openstreetmap.org/) and has 60 277 vertices and 66 895 edges. In step 2, we use "primary" as the minimum road level for selection (https://wiki.openstreetmap.org/wiki/Key:highway). The GPS trajectories were provided by HERE Technologies (https://www.here.com/). As we are looking for patterns in mobility, we used only trajectories with a length of at least 10000m that fall within the bounding box of the network. Trajectories partially within the bounding box were split and each part was treated as a separate trajectory. This left us with 3 795 trajectories as input.

The BJ dataset covers the metropolitan area of Beijing, China. The network is also obtained from OpenStreetMap and has 77 691 vertices and 132 167 edges. In step 2, we again use "primary" as the minimum road level for selection. The GPS trajectories originate from the open Geolife trajectory dataset by Microsoft [37], constrained to this region. We apply the same filtering step as for HR; this left us with 7 520 trajectories as input. The result of our pipeline is shown in Fig. 10. The Geolife dataset mainly contains trajectories obtained from taxi-drivers. We can clearly see in the heat-map of the dataset that there is a high concentration in the top-left, and in our schematic we see that relatively small, loop-like routes, capture a lot of the traffic in that region. Moreover, we see that most bundles occur in pairs, that is, two roughly identical bundles but in opposite directions. Our schematic map generally captures the outer ringroad, but struggles somewhat to capture the grid-like structure of the inner city. This is because the arc-based nature of our schematization algorithm is somewhat opposite to such structures. Future work may investigate hybrid approaches to schematizing such mixed networks of ring roads and non-grid-like structures with arcs, but grid-like structures with parallel segments.



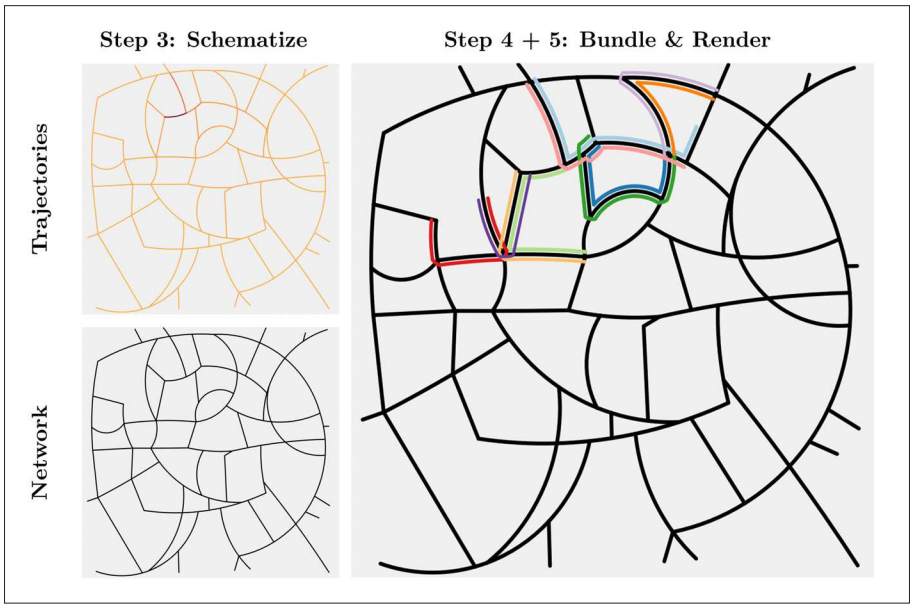


Figure 10. Our pipeline on the BJ data. The input trajectories are shown as a density map. For the map-matched routes, we use a orange to red scale to convey low to high traffic volume per edge We compute the bundles in Step 4 with $S_{min} = 150$, $L_{min} = 8000m$, $p = 0$ and shrunk edge lengths

9. Discussion and Future work

We propose a coordinated pipeline to create abstract visual summaries of mobility patterns in trajectory data. Our proof-of-concept implementation shows that the pipeline is feasible and can fully automatically compute such schematic maps. The advantage of a pipelined approach is that we may improve upon steps individually to improve the eventual result.

Below, we reflect on our choices in the pipeline design, and discuss future work.

Step order. We could in principle also map-match (step 1) after selection (step 2). However, this would not allow for a data-driven approach for selection. Moreover, this forces all traffic to the selected roads, which may hide information on traffic that does not follow the selected (major) roads. For HR, the data-driven selection seems to not have made a large impact on the selected network; for BJ more extensive parts of the network were added because of the data-driven selection. We leave to future work to investigate whether inversion of these steps is able to provide meaningful visualizations.

Similarly, we may wonder whether we would want to detect bundles (step 4) before schematization (step 3). As schematization reduces the network complexity, it is more efficient to do afterwards. It may distort distances, but we can keep track of the original distances if desired – note that aggregation in step 3 does not make each route in the schematic network map to precisely one route in the original network. Another reason supporting our choice for the given order is that schematization may further aggregate dense areas of the network. By bundling afterwards, the support for such bundles grows since they are effectively representing more traffic that generally traverses the dense area in roughly the same way. We believe that our choice helps in promoting spatial diversity, as dense areas with low traffic per road may reduce to a single road with higher traffic. In light of our very spatially uneven datasets, this seems desirable. However, we leave the full exploration of the impact of this choice as future work.

Augmenting the schematic map. We split the map-matched trajectories according to whether or not the route is on the selected network. This leaves us with parts of the trajectories that go through unselected parts of the network and are thus dropped from the schematic map. We intend to explore ways of visualizing these dropped subroutes to provide information on the traffic not part of the selected network. On a computational level, an approach we see for this is to track these subroutes relative to the faces of an embedding of the network. This, however, demands that we meticulously keep track of what happens to these faces during the simplification stages. But it also requires visual design: what do we want to show of

these dropped routes, and how does that combine with the shown bundles? While our mapping between simplifications is discrete in nature, an interesting direction of research would be to extend this to continuous mappings, where routes are also allowed to start in the middle of edges. An appropriate map-matcher should also be selected, since the Fréchet map-matching approach maps only to full edges, though we expect the overall impact of allowing continuous routes here to be minor, as it is performed on the original, detailed network. The primary question is how we can alter the schematization step to allow for high-quality continuous mapping through aggregation and simplification of the network.

Using the Schematic Map. In our proposed pipeline, we do not incorporate the time component of the input trajectories explicitly. Given that the schematic shows strongly aggregated data in a concise way, we can easily use our approach to show small-multiples for different time frames in the data set. This leaves open the question what the best selection method of the road-network is in this case, which we defer to future work.

Our method hides any of the effects of sampling and noise in the data, as well as the deformation and aggregation that occurs in our pipeline, which may be unintuitive to end users. Though the schematic appearance aims to implicitly convey such information, it may be communicated more explicitly with additional uncertainty visualization, albeit at the cost of added visual complexity.

The final schematization is strongly influenced by the selected parameters in steps of our pipeline. We scale parameters by a typical size (e.g. bounding box diagonal or area) to be able to assign parameters independent of scale. Nevertheless, visualizing the impact of the parameters on the end result could help users pick appropriate values for their use cases.

References

[1] Adrienko, N., Adrienko. G. (2010). Spatial generalization and aggregation of massive movement data. *IEEE TVCG*, 17(2) 205–219. doi:10.1109/TVCG.2010.44.

[2] Alt, H., Efrat, A., Rote, G., Wenk., C. (2003). Matching planar maps. *Journal of Algorithms*, 49 (2) 262–283, 2003. doi:10.1016/S0196-6774(03)00085-3.

[3] Andrienko, G., Andrienko, N., Chen, W., Maciejewski, R., Zhao, Y. (2017). Visual Analytics of Mobility and Transportation: State of the Art and Further Research Directions. *IEEE Transactions on Intelligent Transportation Systems*, 18 (8) 2232–2249. doi:10.1109/TITS.2017.2683539.

[4] Arnold, M., Ohlebusch, E. (2011). Linear time algorithms for generalizations of the longest common substring problem. *Algorithmica*, 60 (4) 806–818. doi:10.1007/s00453-009-9369-1.

[5] Buchin, K., Buchin, M., van Kreveld, M., Speckmann, B., Staals. F. (2013). Trajectory grouping structure. In *WADS*, pages 219–230. doi:10.1007/978-3-642-40104-6_19.

[6] Buchin, K., Driemel, A., Gudmundsson, J., Horton, M., Kostitsyna, I., Löffler, M., Struijs. M. (2019). Approximating (k, l)-center clustering for curves. In: *Proceedings SODA*, pages 2922–2938. doi:10.1137/1.9781611975482.181.

[7] Cabello, S., de Berg, M., van Kreveld, M. (2005). Schematization of networks. *Computational Geometry*, 30(3):223–238, 2005. doi:10.1016/j.comgeo.2004.11.002.

[8] Chao, P., Xu, Y., Hua, W., Zhou. X. (2020). A survey on map-matching algorithms. In *ADC*, pages 121–133, 2020. doi:10.1007/978-3-030-39469-1_10.

[9] Chen, J., Hu, Y., Li, Z., Zhao, R., Meng. L. (2009). Selective omission of road features based on mesh density for automatic map generalization. *IJGIS*, 23 (8) 1013–1032, 2009. doi: 10.1080/13658810802070730.

[10] Chen, W., Guo, F., Wang. F.-Y. (2015). A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems*, 16 (6) 2970–2984. doi:10.1109/TITS.2015.2436897.

[11] Garey, M., Johnson, D., Miller, G., Papadimitriou. C. (1980). The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1 (2), 216–227. doi: 10.1137/0601025.

[12] Imai, H., Iri, M. (1986). Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36 (1), 31–41. doi:10.1016/S0734-189X(86)80027-5.

[13] Kraak, M.-J. (2003). The Space-Time cube revisited from a Geovisualization perspective. In *Proc. ICC*, pages 1988–1996.

[14] Kulik, L., Duckham, M., Egenhofer, M. (2005). Ontology-driven map generalization. *Journal of Visual Languages & Computing*, 16 (3) 245–267. doi:10.1016/j.jvlc.2005.02.001.

[15] Lampe, O., Hauser. H. (2011). Interactive visualization of streaming data with kernel density estimation. In *Proc. IEEE PacificVis*, pages 171–178. doi:10.1109/PACIFICVIS.2011. 5742387.

[16] Lee, J.-G., Han, J., Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In *Proc. 2007 ACM SIGMOD*, pages 593–604. doi:10.1145/1247480.1247546.

[17] Meulemans. W. (2014). *Similarity Measures and Algorithms for Cartographic Schematization*. PhD thesis, *TU Eindhoven*, doi:10.6100/ir777493.

[18] Nath, A., Taylor, E. (2020). k-Median clustering under discrete Fréchet and Hausdorff distances, 2020. arXiv:2004.00722.

[19] Scheepens, R., Willems, N., van de Wetering, H., van Wijk, J. (2011). Interactive visualization of multivariate trajectory data with density maps. In IEEE PacificVis, pages 147–154, 2011. doi:10.1109/PACIFICVIS.2011.5742384.

[20] Šuba, R., Meijers, M., van Oosterom, P. (2016). Continuous road network generalization throughout all scales. *ISPRS Journal of Geo-Information*, 5 (8) 145, 2016. doi:10.3390/ijgi5080145.

[21] Thomson, R. (2006). The 'stroke' concept in geographic network generalization and analysis. In Progress in Spatial Data Handling, pages 681–697. Springer, 2006. doi:10.1007/3-540-35589-8_43.

[22] Tominski, C., Schumann, H., Andrienko, G., Andrienko. N. (2012). Stacking-based visualization of trajectory attribute data. *IEEE TVCG*, 18 (12) 2565–2574. doi:10.1109/TVCG.2012.265.

[23] van de Kerkhof, M., Kostitsyna, I., van Kreveld, M., Löffler, M., Ophelders, T. (2020). Routepreserving road network generalization. *In*: Proceedings ACM SIGSPATIAL, pages 381–384, 2020. doi:10.1145/3397536.3422234.

[24] van den Elzen, S., van Wijk. J. (2014). Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE TVCG*, 20 (12), 2310–2319. doi:10.1109/TVCG.2014.2346441.

[25] van Dijk, T., van Goethem, A., Haunert, J.-H., Meulemans, W., Speckmann. B. (2014). Map schematization with circular arcs. In GIScience, pages 1–17. doi:10.1007/978-3-319-11593-1_1.

[26] van Goethem, A., Meulemans, W., Reimer, A., Haverkort, H., Speckmann. B. (2013). Topologicallysafe curved schematisation. *The Cartographic Journal*, 50 (3), 276–285. doi:10.1179/1743277413Y.0000000066.

[27] van Kreveld. M. (2001). Smooth generalization for continuous zooming. In IGU, pages 2180–2185, 2001. URL: https://icaci.org/files/documents/ICC_proceedings/ICC2001/icc2001/file/f13042.pdf.

[28] van Oosterom. P. (2005). Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest. *Cartography and Geographic Information Science*, 32 (4) 331–346. doi:10.1559/152304005775194782.

[29] Weiss, R., Weibel, R. (2014). Road network selection for small-scale maps using an improved centrality-based algorithm. *Journal of Spatial Information Science*, (9) 71–99, doi:10.5311/JOSIS.2014.9.166.

[30] Wolff, A. (2007). Drawing subway maps: A survey. Informatik-Forschung und Entwicklung, 22 (1), 23–44, 2007. doi:10.1007/s00450-007-0036-y.

[31] Wood, J., Dykes, J., Slingsby, A. (2010). Visualisation of Origins, Destinations and Flows with OD maps. *The Cartographic Journal*, 47 (2) 117–129. doi:10.1179/ 000870410X12658023467367.

[32] Wu, H.-Y., Niedermann, B., Takahashi, S., Nöllenburg, M. (2019). A survey on computing schematic network maps: The challenge to interactivity. In: Proceedings Schematic Mapping Workshop. URL: https://www.cg.tuwien.ac.at/research/publications/2019/wu-2019-smw/.

[33] Yang, B., Luan, X., Li, Q. (2011). Generating hierarchical strokes from urban street networks based on spatial pattern recognition. *IJGIS*, 25 (12) 2025–2050. doi:10.1080/13658816. 2011.570270.

[34] Yang, Y., Dwyer, T., Goodwin, S., Marriott, K. (2016). Many-to-many geographically-embedded flow visualisation: An evaluation. *IEEE TVCG*, 23 (1) 411–420, 2016. doi:10.1109/TVCG. 2016.2598885.

[35] Yu, W., Zhang, Y., Ai, T., Guan, Q., Chen, Z., Li, H. (2020). Road network generalization considering traffic flow patterns.

IJGIS, 34 (1) 119–149. doi:10.1080/13658816.2019.1650936.

[36] Yuan, G., Sun, P., Zhao, J., Li, D., Wang, C. (2017). A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47 (1) 123–144. doi:10.1007/ s10462-016-9477-7.

[37] Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. In Proc. WWW, pages 791–800. doi:10.1145/1526709.1526816.