Eurographics Conference on Visualization (EuroVis) 2019
M. Gleicher, H. Leitte, and I. Viola
(Guest Editors)

Volume 38 (2019), Number 3

Route-Aware Edge Bundling for Visualizing Origin-Destination Trails in Urban Traffic

W. Zeng[1] , Q. Shen[2†], Y. Jiang[2], A. Telea[3]

[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[2]The Hong Kong University of Science and Technology, Hong Kong, China
[3]University of Groningen, the Netherlands

Abstract

Origin-destination (OD) trails describe movements across space. Typical visualizations thereof use either straight lines or plot the actual trajectories. To reduce clutter inherent to visualizing large OD datasets, bundling methods can be used. Yet, bundling OD trails in urban traffic data remains challenging. Two specific reasons hereof are the constraints implied by the underlying road network and the difficulty of finding good bundling settings. To cope with these issues, we propose a new approach called Route Aware Edge Bundling (RAEB). To handle road constraints, we first generate a hierarchical model of the road-and-trajectory data. Next, we derive optimal bundling parameters, including kernel size and number of iterations, for a user-selected level of detail of this model, thereby allowing users to explicitly trade off simplification vs *accuracy. We demonstrate the added value of RAEB compared to state-of-the-art trail bundling methods on both synthetic and real-world traffic data for tasks that include the preservation of road network topology and the support of multiscale exploration.*

CCS Concepts

•***Human-centered computing*** $\rightarrow$ ***Graph drawings; Geographic visualization;***

1. Introduction

Movement can be defined as change of an object's position or geometric attributes over time [DWL08]. For a specific time period, the movement of an object can be modeled with an origin (O), a destination (D), and consecutive positions (trail) in-between. Due to fast development of location sensing technologies, massive amounts of OD trails, such as vessel movements and taxi trips, have been collected. Studies of OD trail data have revealed many movement patterns and contributed to many applications, *e.g.*, transportation planning [WHB*12, SWvdW*12, SHvdW*16].

Visualizing OD trails is a hot and challenging topic. Conventional methods that connect origins to destinations with lines, or else plot the actual trails, can easily cause visual clutter. To tackle this problem, trails can be aggregated into flows. Several such techniques have emerged, such as intelligent routing layouts [PXYH05, VBS11] and spatial mapping [AA11, GZ14]. Such methods can reveal overall traffic patterns to answer questions such as: What are popular roads in a city? Where do people come from and head to? Yet, many such methods ignore information of individual OD trails, such as the interplay of trail parameters like trail length, flow (number of vehicles) along a trail, and how trails correspond to roads having given importances in a city (*e.g*, highways or secondary roads). Another limitation is that high-level aggregation does not allow one to follow individual trails. While, formally speaking, aggregation will result in the latter problem, in certain scenarios, *e.g.*, finding abnormal movements, additional analytics are required to complement such visualizations to enable users to follow, at least partially, individual trails.

A particular class of methods produce simplified views of large trail datasets by grouping trails into *bundles*. Simply put, bundling aims to strike a balance between *aggregation* (to yield a simplified view) and *details* (to enable one to follow, at least partially, individual trails) [LHT17b]. Many bundling methods exist, such as force-based [HvW09, SHH11], image-based [HET12, LHT17a], and geometry-based [Hol06, CZQ*08]. Most such methods however do not consider *spatial* constraints on the emerging bundles. In a city, objects move along roads, and events of interest like traffic jams and accidents also happen along roads. Hence, incorporating such spatial constraints in the bundling is of crucial importance. Simply put, if urban OD trails follow roads, so should also the simplified bundles do.

We address the above issues by proposing a new Route-Aware Edge Bundling (RAEB) method. Our contributions are as follows:

- RAEB extends a state-of-the-art generic bundling method (KDEEB [HET12]), well known for its speed and simplicity, to incorporate specific constraints of urban OD trails. To our knowledge, this is the first adaptation of bundling that accounts for *spatial* trail constraints, a task that is prominently listed on the open research agenda for bundling research [LHT17b].
- RAEB also allows one to select how to trade off *simplification* (how much to bundle) *vs accuracy* (how much to respect the underlying road network). To our knowledge, no bundling method offers a similar explicit trade-off, or even discusses it;

† Q. Shen is the corresponding author. E-mail: qshen@ust.hk

- RAEB addresses in a principled, and automatic, way the so far not studied problem of selecting suitable parameters for KDEEB such as kernel radius and number of iterations, thereby making bundling simpler to use.

Experiments on synthetic and real-world urban traffic data show that RAEB generates more realistic, and closer to ground-truth results, than KDEEB, but preserves KDEEB's simplicity and scalability.

This paper is structured as follows. Section 2 presents related work on (constrained) OD trail bundling. Section 3 outlines the main limitations of state-of-the-art (KDEEB-like) bundling methods for our problem context. Section 4 details the OD trail analysis we perform to support our constrained bundling. Sections 5 and 6 introduce RAEB. Section 7 shows results of RAEB on synthetic and real-world urban trail datasets. Section 8 discusses our proposal. Finally, Section 9 concludes the paper.

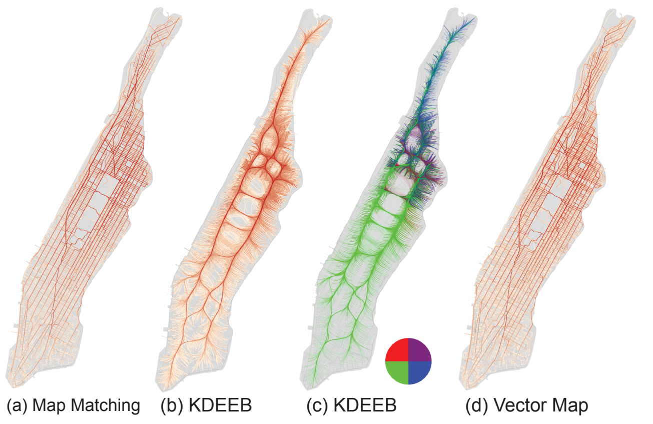


Figure 1: *Methods for visualizing urban-traffic OD trails: (a) density maps; (b,c) KDEEB bundles colored by density and direction, respectively; (d) vector maps. See Sec. 2.*

2. Related Work

We group related work in five categories: *Urban traffic visual analytics* (discussing general methods to visually analyze urban traffic data); *OD trails visualization* (discussing more specific methods to visualize OD trails); *map matching* (discussing how recorded vehicle trails can be constrained to a 'ground truth' road network); *trail bundling* (introducing techniques related to bundling, which is the approach we choose for urban traffic visualization in our work); and *constrained bundling* (which refines the earlier point in discussing how bundling techniques can use extra data to constrain their outputs).

Urban Traffic Visual Analytics: Many visual analytics tools have been developed to explore urban traffic data and to understand urban dynamics and human activities, aiming to enhance traffic management and assessment. Systematic reviews are presented in [CGW15, AAC*17]. Specific techniques to study OD urban traffic trails include advanced indexing methods [FPV*13, ZFMA*16] and new interaction models [KTW*13, ZFMA*16]) aiming to help movement queries. After filtering OD trails, such techniques are typically complemented with statistical analysis supported by statistical graphics. Typical methods in this class use a traffic density map atop the road network. Figure 1(a) shows an example for a Manhattan taxi-trips dataset. This method naturally renders traffic data where it occurs (on the road network) [KSBE18], which is not the case with bundling methods (discussed next below). Yet, density maps do not offer a simple-to-use way to explore data in a *multiscale* way. That is, there is no *automated* way to 'gather' the traffic data and render it on fewer roads, for instance, if one needs a simplified visualization.

OD Trails Visualization: Urban movement data, such as taxi trips and mobile phone traces, is commonly modeled as OD trails containing information of origin, destination, and optionally in-between locations. The many existing OD visualization techniques can be categorized into matrix-based and trail-based. *Matrix-based* techniques show ODs as an adjacency matrix, with cell colors showing flow magnitudes between OD pairs. Sorting rows and columns can reveal cluster patterns [WF09]. However, matrix techniques cannot show actual spatial data (in-between locations). To address this, OD maps [WDS10] divide a traditional OD matrix into two layers showing origins and destinations, respectively. *Trail-based* techniques intuitively show the trail spatial data using node-link metaphors. Several such methods are also known under the name flow-based methods [PXYH05]. However, we prefer the term trail-based methods, since flow is more commonly associated with using vector and tensor field methods to aggregate and visualize geospatial movement [FKSS13, KJW*18]. Trail-based methods face visual scalability issues, as clutter and excessive edge crossings or overdraw appear for even moderate-size trail sets. To mitigate this, trails can be spatially aggregated by graph partitioning [GZ14], density-based clustering [vLBR*16, WvdWvW09], continuous flow map layouts [PXYH05], and composite layouts [CKS*16]. While both matrix-based and trail-based methods perform well in reducing clutter, information of *individual* OD trails is not preserved. This may cause analysis problems when exploring urban traffic data – for example, an abnormal suspect movement to a remote place can be grouped together with normal trip trails.

Map Matching: Huge amounts of position data are available these days, benefiting many applications such as traffic analysis and management. Yet, measurement errors caused by location-sensing devices and sampling uncertainty introduced by the devices' sampling rates can yield imprecise vehicle positions. Map matching addresses such errors by aligning vehicle positions with road network data, based on the assumption that a vehicle is constrained to move along a finite, given, road network [QON07]. For OD traffic data, graph theoretic methods that use spatial metrics such as distances to, and intersection angles of, the road network, can be applied to perform map maching. For traffic data consisting of origins, destinations, and in-between recorded positions, map matching approaches exist that take the entire trail (sequence of positions) into account, or alternatively use faster iterative methods that only consider parts of such sequences [BPSW05]. However, all map-macthing methods will introduce errors into the matched data. Krüger *et al.* [KSBE18] propose an interactive interface for visually examining errors and their effects.

The first step of map matching is to find a list of candidate nodes on the road network that are within a certain radius of each recorded trail position. For a given radius, dense road networks can return many candidate nodes and thus generate more accurate map matches [Qud06]. Simplifying a given road network to next map trails to main roads only will cause undesirable errors. Our method 'blends' map matching advantages (first, constraining trails to routes, within user-specified simplification bounds of the road network) with bundling advantages (creating smoothed trails that are easy to visually follow end-to-end [LHT17b]).

Trail Bundling: A different trade-off between clutter reduction *vs* showing individual trails is proposed by trail or edge bundling methods. These methods group spatially close, and optionally data-related, edges in a graph drawing or arbitrary 2D or 3D trails (curves). Edge bundling and trail bundling are closely related [LHT17b], so we next refer to them both as bundling. Following recent surveys [ZXYQ13, LHT17b], bundling methods can be grouped in three classes: geometry-based, force-based, and image-based, as follows. *Geometry-based* methods cluster trails using a skeleton-like control mesh that specifies how similar edges are routed. Such methods differ mainly in how they construct control meshes, *e.g.*, using hierarchical graph drawing [Hol06], triangulation [ZYC*08, LBA10b], complex polygons [CZQ*08], and functional decomposition [HPNT18]. Control meshes make geometry-based methods flexible. Yet, building such control structures can be (very) slow for large trail sets. *Force-based* methods remove the need to compute an explicit control structure by modeling interaction between spatially close trails as a force field [HvW09]. While conceptually simpler than geometry-based methods, force-based methods still are expensive – they cannot treat more than a few thousand trails at interactive rates. *Image-based* techniques take both the control structure idea of geometry-based methods (*e.g.*, skeletons [EHP*11]) and the field model of force-based methods to scalable levels by implementing them via image processing on the GPU. Current state-of-the-art bundling methods fall into this class. These use the gradient of the kernel density estimation (KDE) of the trail set as bundling force field (KDEEB [HET12], CUBu [vdZCT16], FFTEB [LHT17a]), and can bundle millions of trails at interactive rates. Figures 1(b) and (c) show KDEEB bundling of the Manhattand taxi trip dataset with bundles colored to show density, respectively directions. While such results show a strong simplification as compared to the original density map (Fig. 1(a)), bundles largely *ignore* the road structure, which is not desirable.

Constrained Bundling: Specialized methods have been proposed to bundle data with specific spatial constraints. These include minimizing ambiguities in visually following O-to-D connections [LLCM12, BRH*17]; separating trails having different directions [SHH11, PHT15]; and bundling specific types of data such as paths constrained to a 3D curved surface [LBA10a] and connection paths in the human brain [BSL*14, YSD*17, HPNT18]. Closely related to our scope, KDEEB [HET12] presents an experiment where bundles are repelled by so-called obstacles by a force field equal to the obstacles' distance transform gradient. While this idea makes bundles *avoid* simple-shaped spatial obstacles, it cannot be readily used to constrain bundles to *follow* spatial sites like roads. At the other end of the spectrum, vector maps [TP15] use road networks as control skeletons for B-spline-based trail bundling. This method can effectively route bundles to avoid 3D landscape-like obstacles, but offers only marginal simplification when bundling urban traffic trails on dense road networks. Figure 1(d) shows a vector map for the Manhattan taxi trip dataset. The result is very close to the original density map (Fig. 1(a)), and quite far from the high simplification produced by KDEEB-type methods (Figs. 1(b,c)). Simply put, for dense road networks, vector maps simplify too *little*, since they aim to obey the roads too much; at the other extreme, KDEEB-like methods simplify too *much*, as they ignore such road constraints. We show next how we can strike the desired balance between these two extremes.

3. Problem Statement and Solution Overview

We want to use bundling to simplify urban trails. Since KDEEB is one of the state-of-the-art existing bundling methods (Sec. 2), we propose to extend and adapt it to our specific goal. We next introduce KDEEB (Sec. 3.1), outline its main limitations for urban trail bundling (Sec. 3.2), and finally outline our proposal (Sec. 3.3).

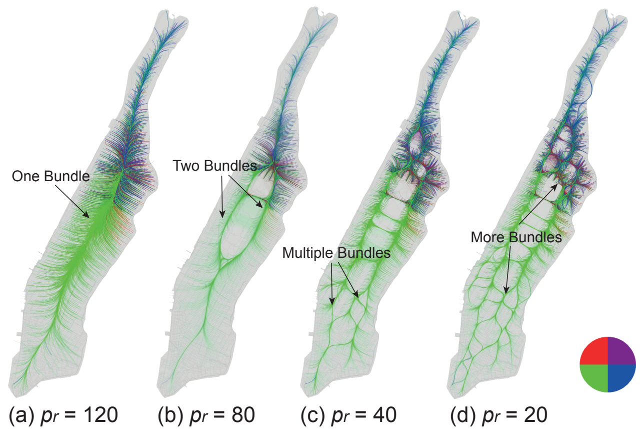


Figure 2: *KDEEB applied to Manhattan taxi trips with different kernel sizes p_r: (a) 120, (b) 80, (c) 40, and (d) 20. See Sec. 3.2.*

3.1. KDEEB Algorithm

KDEEB bundles arbitrary trails (2D curves) in four steps [HET12]: sampling, density gradient estimation, advection, and smoothing. KDEEB starts with a set of trails $\mathbf{e}_i$. Each such trail $\mathbf{e}_i$ is first uniformly sampled into a sequence of (roughly) equally-spaced points $\mathbf{x}^i_1, \ldots, \mathbf{x}^i_n$, given a user-supplied sampling step σ. Next, a density map $\rho : \mathbb{R}^2 \rightarrow \mathbb{R}^+$ is computed from the set of all sampling points of all trails $D = \{\mathbf{x}^i_j\}$ as

$$\rho(\mathbf{x} \in \mathbb{R}^2) = \sum_{\mathbf{y} \in D} K\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{p_r}\right) \tag{1}$$

where K is a 1D Epanechnikov (parabolic) kernel of radius p_r. Next, each sample point $\mathbf{x} \in D$ is advected to a new position $\mathbf{x}'$ in the normalized gradient of ρ by

$$\mathbf{x}' = \mathbf{x} + p_r \frac{\nabla\rho}{||\nabla\rho||}. \tag{2}$$

Finally, the sampled trails are smoothed by Laplacian filtering to remove jitters created by the discrete advection (Eqn. 2). To generate tight bundles, all above steps are repeated p_n times, while decreasing the kernel radius p_r by a fixed decay ratio $\lambda \in [0.5, 0.9]$ at each iteration to ensure convergence.

3.2. Problem Identification

We identify the following problems when applying KDEEB to bundle road-constrained trails.

Road neglect (P1): Urban OD trails follow roads, so bundles should also respect roads as much as possible (Sec. 2). KDEEB, but also most if not all other bundling methods, are not designed with such constraints in mind, so they produce bundles that are far away from, or cutting across, the road structure, especially at coarse scales (see *e.g.* Fig. 2 and Sec. 7.1 further). This makes the bundled visualization

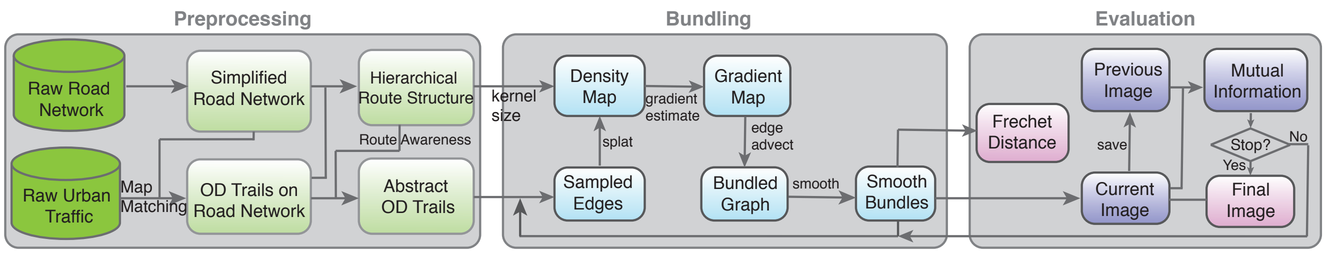


Figure 3: *Overview of RAEB pipeline. RAEB consists of three phases: Preprocessing creates an initial control structure; bundling does the actual trail grouping; and evaluation determines when to stop bundling. See Sec. 3.3.*

less useful since it is hard to mentally map the original OD trails to the shown bundles.

Multiscale exploration (P2): KDE-type methods use the kernel radius p_r to control the creation of multiscale bundles. Small p_r values preserve more trail details, but fail showing the trail-set's overall structure; large p_r values strongly simplify, but deform the bundled trails too much. We show this next for the Manhattan taxi dataset. Following [vdZCT16, HET12], we set p_r to 5% of the drawing size (720×1440 pixels) to obtain $p_r = 80$ pixels (Fig. 2(b)). Clearly, this setting deforms the trail set too much. Smaller p_r values yield more locally-detailed bundles (Figs. 2(c,d)) but low simplification. The question is, thus, how to choose a good p_r value. An extra problem is that road networks *and* road-traffic density *vary spatially*, and urban maps can have arbitrarily complex shapes (see also the Shenzhen taxi data in Sec. 7.3). Hence, how to set p_r to obtain the desired balance between simplification and road following?

Quality control (P3): Image- and force-based bundling methods work iteratively. Different methods recommend different numbers of iterations p_n, *e.g.*, 10 for KDEEB [HET12] and 15 for CUBu [vdZCT16]. However, p_n is related to how the initial KDE radius p_r is decreased over time and also to other parameters such as the sampling step σ [vdZCT16]. We need a better understanding of these processes to guarantee the quality of the desired bundles. Separately, since bundling deforms our spatial trails, we would like to have an objective measure to capture the amount of deformation.

3.3. RAEB Overview

To alleviate the problems P1, P2, and P3 outlined above, we propose Route-Aware Edge Bundling (RAEB). The RAEB pipeline has three main parts (Fig. 3), as follows.

Preprocessing (Sec. 4): We first build a simplified hierarchical road-and-traffic network representation from the input data using a map matching algorithm. This let us specify a bundling level-of-detail based on a *route awareness* user parameter and also to suitably and automatically set the kernel size p_r.

Bundling (Sec. 5): We apply KDEEB to the hierarchical structure computed in preprocessing. We do not set a 'hard' maximal iteration count p_n, but automatically stop bundling based on a new bundling stability metric that measures the normalized mutual information between two consecutive bundling moments.

Evaluation (Sec. 6): We evaluate both the aforementioned bundling stability metric (to determine when to stop bundling), and also a quality metric that measures the difference between the bundling result and the underlying road network (to determine the overall quality of the produced result).

4. Preprocessing

We next explain the kind of input data that our method works on (Sec. 4.1) and how the data is preprocessed prior to actually performing the bundling (Sec. 4.2).

4.1. Basic Traffic Concepts

Road network: RAEB takes as its first input a graph $G = (V, E)$ that stores the topology and spatial layout of a road network. Graph vertices V denote points along roads, and edges E denote road segments connecting these points. Let $deg(\mathbf{v})$ be the degree of (number of edges connected to) a vertex $\mathbf{v} \in V$. We classify $\mathbf{v}$ as an endpoint if $deg(\mathbf{v}) = 1$; a road midpoint if $deg(\mathbf{v}) = 2$, or a crossroad, if $deg(\mathbf{v}) > 2$, respectively. Let V' be the union of endpoints and crossroads in V. Let a route $r = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ be a sequence of n consecutive vertices where $\mathbf{v}_1$ and $\mathbf{v}_n$ are endpoints or crossroads in V', while all other $\mathbf{v}_i$ are midpoints from V. We next use the simplified graph $G' = (V', E')$ in all our computations instead of G. For simplicity, we next denote G' as G.

Urban traffic: RAEB's second input is a raw urban traffic dataset, which can have one of the following two forms:

- *UT-1*: Origin $\mathbf{x}_o$ and destination $\mathbf{x}_d$ locations only, such as the New York taxi trips [FPV*13] and Singapore public transportation rides [ZFMA*16] datasets.
- *UT-2*: A sequence of GPS positions, see *e.g.* the Stuttgart scooter [KTW*13] and Hangzhou taxi trips [WCW*14] datasets.

Both above data types can be seen as a set of 2D trails $D = \{\mathbf{e}_i\}$ (following notations in Sec. 3.1). UT-1 trails are lines $\mathbf{e} = (\mathbf{x}_o, \mathbf{x}_d)$; UT-2 trails are n-point polylines $\mathbf{e} = (\mathbf{x}_o, \ldots, \mathbf{x}_d)$, respectively.

We next use G and D to compute a simplified, hierarchical, model of urban traffic, in three steps – map matching (Sec. 4.2), hierarchical road structure construction (Sec. 4.3), and trail abstraction (Sec. 4.4).

4.2. Map Matching

We first constrain raw movements to the underlying road network, *i.e.*, map D to G. We need this for two reasons: First, recorded locations in D have errors due to imprecise GPS localization, while vertex locations V in the road network are precise. Secondly, D is typically much more complex than G: For example, the New York traffic data [NYC] has millions of taxi trips (trails) per day, resulting in tens of millions of locations in D; yet, the corresponding simplified road network has under 100K routes.

Many map matching methods exist, *e.g.*, shortest paths, minimum turns, and ST-matching [LZZ*09] for GPS traces. Here, we employ the following matching methods:

UT-1 trails: Given such a trail $\mathbf{e} = (\mathbf{x}_o, \mathbf{x}_d)$, we first find closest vertices $\mathbf{v}_o$ and $\mathbf{v}_d$ in V for $\mathbf{x}_o$ and $\mathbf{x}_d$, respectively. Next, we find the shortest path in G between $\mathbf{v}_o$ and $\mathbf{v}_d$. Shortest paths can store pre-computed paths to further accelerate computation. Though the algorithm may not find a unique 'true' trajectory, it gives a good estimation for a large dataset [Som14], as the NYC taxi (see next Sec. 7.2). Importantly, we do not aim (or claim) to be able to do better than existing methods in estinmating actual vehicle paths. This is a hard problem in itself, as explained in [Som14]. Rather, our goal is that, for a *given* dataset, we aim to produce a simplification that balances well between ground-truth (information present in the raw data) and visual complexity (ease to read the final visualization).

UT-2 trails: Given a trail $\mathbf{e} = (\mathbf{x}_o, \dots, \mathbf{x}_d)$, we use the ST-matching algorithm, which considers not only spatial properties of road network, but also temporal constraints [KSBE18]. This method is preferable for sparse GPS traces, such as the Shenzhen taxi data (Sec. 7.3).

For each raw trail $\mathbf{e} \in D$, both above methods return a sequence of one or more routes $(r_1, \dots, r_n)$ with $r_i \in E$. We next replace the raw trail $\mathbf{e}$ by the trail $(\mathbf{x}_o, r_1, \dots, r_n, \mathbf{x}_d)$, *i.e.*, essentially 'snap' the raw trails to routes in the map. This way, all trails next use only (accurate) map locations in V.

4.3. Hierarchical Route Structure Construction

To compute a simplified visualization of urban data, we need to define how elements (trails, routes) should be aggregated. To this end, we compute the *importance* of a route and its trails assigned by map matching, and next simplify to keep only the most important items. A route's importance should capture *both* the route's geometry and its traffic. Indeed: If we used only route geometry, we may inadvertently merge important traffic flows that run on spatially close or otherwise similar routes. Conversely, if we used only traffic data, we would inadvertently merge flows that correspond to different routes, like in standard bundling. We define a route's importance by the following attributes:

Route length ($len(r)$): Longer routes are arguably more important, thus should appear more saliently in a simplified traffic visualization. A first argument to this end is that, if we want to simplify such a visualization, then shorter routes (which take less visual space to display) are a first candidate for elimination. This design is used by many route visualizations that show, at a coarse level, only the longest routes on a given map. A second argument relates to our usage of bundling: Prior work has shown that bundling too many short trails creates a high amount of clutter [HvW09, vdZCT16]. In other words, if we aim to use bundling as a simplification technique, then we should reduce the number of (too) short trails that get bundled. Of course, certain short routes can be very important and should not be simplified away. The road hierarchy and flow magnitude parameters (discussed next) take care of this. Technically, we compute a route's length $len(r)$ as the sum of its segment lengths $\|r_i\|$, normalized by the length of the longest route in V.

Road hierarchy ($hier(r)$): Urban roads are typically hierarchical, *e.g.*, freeways, arterials, collectors, and local roads [WCW*14]; or trunk, primary, secondary, and tertiary [OSM]. Higher levels indicate faster traffic speed and less access to residential properties, and are more important to keep in a simplified visualization than less important levels. We quantify a route's importance on a four-level ordinal scale (Tab. 1).

Category	**OSM indicator**	**Score** $hier(r)$
Expressway	motorway, trunk	1
Trunk road	primary, motorway_link	0.75
Secondary road	secondary, tertiary	0.5
Branch road	unclassified, residential	0.25

Table 1: *Hierarchy scores for corresponding route types and representative OSM indicators.*

Flow magnitude ($flow(r)$). While $len(r)$ and $hier(r)$ measure the importance of a route in the road network, measuring how important a route is to actual traffic is also needed. For this, we consider a route's flow, *i.e.*, how many trails in D pass through r, normalized to the highest flow value over D.

We compute the importance of a route as $I(r) = w_{len}len(r) + w_{hier}hier(r) + w_{flow}flow(r)$. We experimentally found that the weights $w_{len} = 0.3$, $w_{hier} = 0.1$, and $w_{flow} = 0.6$ yield a good balance between the three types of a route's importance when bundling urban trails, with flow being highlighted most. The underlying assumption here is that we want to keep high-flow routes salient in our bundled (simplified) visualizations. If one desires to emphasize other route aspects, *e.g.*, length, weights can be easily modified to this end. Next, we sort all routes r descendingly on $I(r)$ and group them into N nested hierarchical levels R^k, $1 \le k \le N$. In practice, we used $N = 5$ levels, each containing the top 5%, 10%, 20%, 40%, and 100% most important routes.

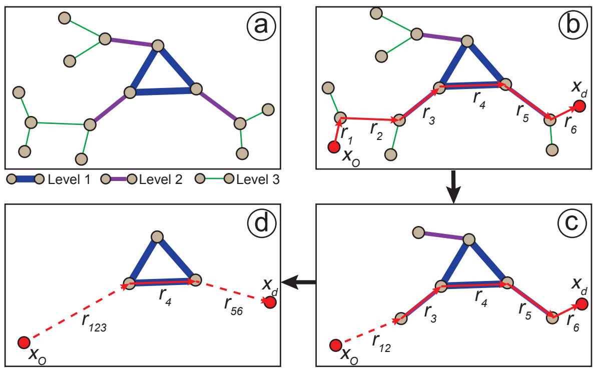


Figure 4: *Hierarchical route structure and OD trail abstraction: (a) Three route levels are constructed (blue, purple and green). A raw OD trail is abstracted on (b) level 3, (c) level 2, and (d) level 1.*

4.4. Trail Abstraction

Road neglect is one of the key problems of generic bundling urban trails (P1, Sec. 3.2). As discussed there, using a global fixed bundling kernel radius p_r does not always give a good balance between simplification and road-following of the trail set. To tackle this, we introduce a *route awareness* parameter (p_{ra}). For a N-level hierarchical road structure (Sec. 4.3), p_{ra} ranges from 0 to N. For $p_{ra} = 0$, trails along are (maximally) abstracted to straight lines; when $p_{ra} = N$, no simplification is done; when p_{ra} is set to a value $0 < k < N$, routes in $R^1, \dots, R^k$ will be used to abstract the trails.

To illustrate this, consider a route hierarchy with $N = 3$ levels (thick blue, medium-thick red, and thin green, see Fig. 4(a)). Figures 4(b-d) show how a trail $(\mathbf{x}_o, r_1, \dots, r_6, \mathbf{x}_d)$ is abstracted on the three hierarchy levels: In image (b), $p_{ra} = 3$, so all routes are preserved; in image (c), $p_{ra} = 2$, so routes r_1 and r_2 are merged together as they are both part of R^3. Finally, in image (d), $p_{ra} = 1$, so only route r_4 is kept.

5. Adapting KDEEB to Urban Trails

We now use the hierarchical data model introduced in Sec. 4.3 to adapt KDEEB [HET12] to better reveal the topology of urban traffic. For this, we propose two modifications of KDEEB: Kernel size setting (Sec. 5.1) and density map generation (Sec. 5.2).

Algorithm 1 KernelSizeSetting

```
Input: Top $N$ routes $P = \{P_1, ..., P_N\}$
Output: Initial kernel size $p_r$
 1: for i = 1 to $N$ do
 2:     for j = i + 1 to $N$ do
 3:         $d$[i][j] = $d$[j][i] = DiscreteFrechetDistance($P_i$, $P_j$)
 4: C = $DBSCAN(P, \varepsilon, minNum)$;
 5: $C_{max}$ = argmax$_{C_i \in C}|C_i|$;
 6: $d_{geo} = 0$;
 7: for each $P_i \in C_{max}$ do
 8:     for each $P_j \in C_{max}$ && $i \neq j$ do
 9:         $d_{geo}$ = $d_{geo}$ + $d$[i][j];
10: $p_r = d_{geo}/|C_{max}|/(|C_{max}| - 1)/2$;
11: return $p_r$
```

5.1. Optimal Kernel Size Setting

KDE-type bundling methods only consider the size of the target image to set the kernel size p_r. As outlined in Sec. 3.2 (P2), this heuristic for setting p_r does not work well for bundling urban trails. Ideally, we want to bundle closely-related trails (*e.g.*, movements on bidirectional roads) but separate loosely-related or unrelated trails (*e.g.*, movements on two different highways). For this, we must consider both image size and geometric properties of the road network.

To this end, we propose an automatic method for setting the initial kernel size p_r (see Alg. 1). We consider the set P of top-N routes in our hierarchy (Sec. 4.3). We compute pairwise distances between each route-pair $(P_i, P_j) \in P \times P$ and use these to group routes into a set of clusters C. For the largest cluster C_{max}, *i.e.* having most routes, we compute the average distance d_{geo} between all routes in C_{max}. Finally, we set the initial kernel size p_r to half d_{geo} (in pixel units). We measure distance between routes by the discrete Fréchet distance, which considers both point locations and ordering. This metric is one of the most popular methods for movement analysis [GLW11], and can be computed efficiently [EM94]. Setting N is done as follows: Too small N values should be avoided as they yield a too sparse initial road-set P, and next a too large kernel p_r, thus too strong bundling. Conversely, too large values N select in P routes that are unrepresentative for the entire road network, yielding too small p_r values. From our experiments, we found that setting N to 1% of the total number of routes in G gives good results. The clustering method is built upon DBSCAN, which has been successfully used to cluster trajectories [LHW07]. We set ε and $minNum$ following the strategies in [LHW07]. This yields $\varepsilon = 5$ pixels $minNum = 8$, which we next used for all results in this paper.

5.2. Density Map Generation

As explained in Sec. 4.4, depending on the user level of detail k, routes are abstracted into actual route segments and artificial straight-line segments (see also Fig. 4). Hence, we must keep bundles close to the road network R^k. A key observation is that bundles can freely follow the artificial line segments in R^k, since these are anyways *abstractions* of the road network geometry. In contrast, bundles *should* follow the actual (real) route segments $R_{aware} \subset R^k$ as much as possible, since these encode actual geometric information. To achieve this, during density map generation (Eqn. 1), we increase the density for pixels on route segments in R_{aware}. For this, we replace ρ by

$$\rho_{raeb}(\mathbf{x} \in \mathbb{R}^2) = \sum_{\mathbf{y} \in D} K\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{p_r}\right) + \theta \sum_{\mathbf{r} \in R_{aware}} \Theta(\|\mathbf{x}-\mathbf{r}\|), \quad (3)$$

where $\Theta \in \mathbb{R}_0^+$ is an indicator function telling if $\mathbf{x}$ is close to, or on, a trail $\mathbf{r} \in R_{aware}$. Θ can be set to the same parabolic kernel K as for classical KDE (Sec. 3.1). However, a setting which is simpler to implement, and slightly faster to compute, and gives results as good as the parabolic one, is $\Theta(x) = 1$ if $x = 0$, else $\Theta(x) = 0$. This is equivalent to 'boosting' the density along R_{aware}. The boosting factor θ should exceed the maximum ρ_{max} of ρ as defined by Eqn. 1. In practice, setting $\theta = 1.1\rho_{max}$ gives the desired results. That is, this guarantees that trail samples close to R_{aware} are attracted to R_{aware} during advection (Eqn. 2).

6. Bundling Evaluation

As explained in Sec. 3.2 (P3), we need to measure the outcome of a bundling iteration to determine when to stop bundling in a more principled way than just using a maximal number of iterations. Separately, we want to measure the overall deformations produced by an entire bundling sequence to a given trail set, to determine if these are acceptable or not in a given context. We address both these tasks next in Secs. 6.1 and 6.2 respectively.

6.1. Bundling Termination Evaluation

For force-based and image-based bundling methods, there is so far no explicit way to set the maximum number of iterations p_n. Setting p_n too low creates too loose bundles. Conversely, setting p_n too high wastes computational resources.

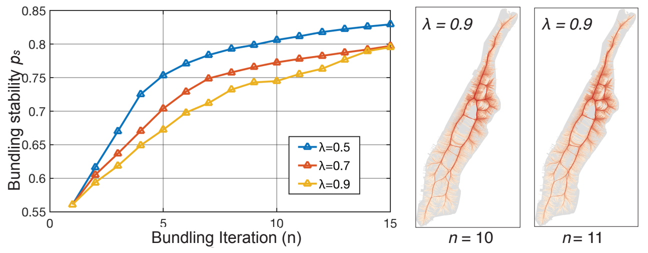


Figure 5: *Left: Bundling stability p_s measured at each bundling iteration for different decay ratios λ. Right: Visually indistinguishable images are generated at iteration 10 and 11 for $\lambda = 0.9$. See Sec. 6.1.*

We replace the *explicit* user setting of p_n by an *implicit* bundling termination criterion. For this, we measure the similarity between the images X and Y produced by two consecutive bundling iterations. When this similarity exceeds a given level, we consider that bundling has objectively converged, and stop the process. To measure image

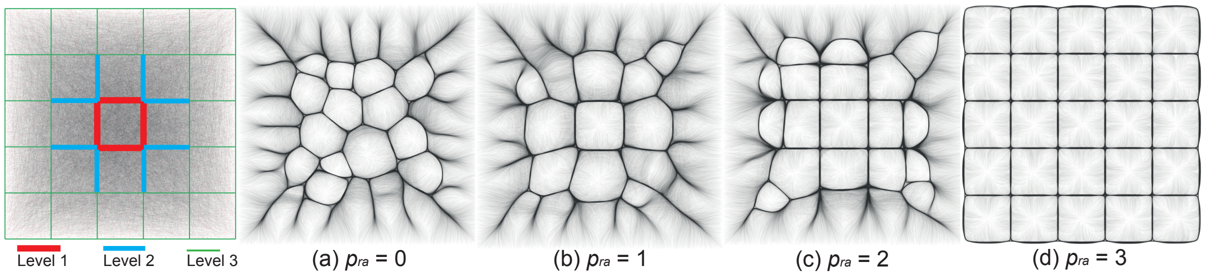


Figure 6: *Leftmost: 100K raw artificial OD trails on a grid road network represented by three hierarchy levels. (a) - (d): RAEB bundling results with route awareness parameter p_{ra} set to $0,\ldots,3$, respectively. See Sec. 7.1.*

similarity, we use Mutual Information (MI), a basic concept from information theory, introduced by Maes *et al.* [MCV*97] to compare medical images. While the original paper required image registration to find corresponding point pairs, we do not need this in our case, since images for all bundling iterations are aligned by construction. Given two such images X and Y, seen as discrete random variables taking intensity values in the range $[0,255]$ (due to the blending of bundled trails), MI is computed as

$$MI(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)log\left(\frac{p(x,y)}{p(x)p(y)}\right), \tag{4}$$

where $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively (computed via normalized intensity histograms), and $p(x,y)$ is the joint probability function of X and Y. Since MI is affected by image size, to obtain a resolution-independent metric, we use the normalized MI defined as

$$NMI(X,Y) = \frac{2MI(X,Y)}{H(X)+H(Y)} \tag{5}$$

where $H(X) = -\sum_{x\in X} p(x)log(p(x)))$ is the entropy of image X. We denote NMI as bundling stability p_s.

Figure 5(left) shows NMI values at each bundling iteration $n \in [0,15]$ for the Manhattan taxi dataset, for three different kernel-size decay values λ (see Sec. 3.1). We see how NMI increases with the iteration count. Also, we see that smaller λ values lead to faster NMI increases. Figure 5(right) shows the images for iterations 10 and 11 ($\lambda = 0.9$). These images are hardly distinguishable from each other, so bundling has stabilized in this case and can be stopped. For this image pair, we find a NMI value of 0.8, which we next use to predict bundling convergence. That is, we bundle until the computed NMI exceeds the threshold value $p_s = 0.8$. We verified that this automatic bundling termination yields stable, converged, results for all our tested datasets and parameter values.

6.2. Bundle Deviation Evaluation

Finally, we would like to evaluate the overall deformation caused by our bundling method, to be able to reason about the amount of changes induced by the bundling simplification (P3, Sec. 3.2). For this, we compute the average Fréchet distance between each bundled trail and its original version in D, *after* the latter was mapped to the road network (Sec. 4.2). We use trails after mapping to the road network since (a) raw UT-1 trails do not anyways contain geometric information, but only origin and destination positions, so computing distances between such straight-line segments and bundles is misleading; and (b) UT-2 trails come from GPS-recorded positions, which can be inaccurate. Thus, 'snapping' raw trails to the road network prior to computing bundling errors provides a better ground-truth for the bundling deviations. Evaluating this deviation (of bundles from road-mapped trails) allows us to see how much the bundling process has distorted the actual data. We discuss this next in Sec. 8.2.

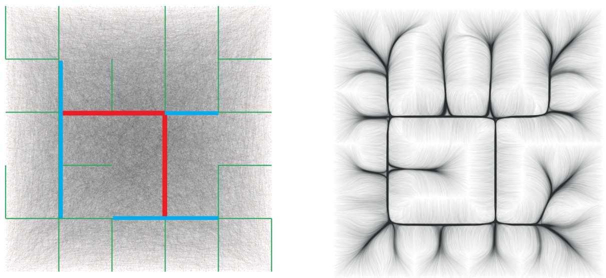


Figure 7: *Left: Raw 100K artificial OD trails on a hierarchical road network. Right: RAEB bundling result with $p_{ra} = 2$. See Sec. 7.1.*

7. Applications

We evaluate RAEB on four datasets. First, we analyze its behavior *vs* its free parameters on synthetic trail-and-road-network data (Sec. 7.1). Next, we consider three real-world urban traffic datasets from New York (Sec. 7.2), Shenzhen (Sec. 7.3), and Manhattan (Sec. 7.4). For all datasets, we outline improvements of our method *vs* KDEEB.

7.1. Synthetic Data

We use two trail datasets D, each with 100K straight-line (UT-1 like) trails with endpoints randomly distributed in a unit square (similar to the dataset used in [HET12], Figure 9(c)) and two reference road networks G defined on an uniform 5×5 grid, with three hierarchy levels. The two trail-sets and corresponding networks are shown in the left images in Figures 6 and 7. We apply RAEB using an image resolution of 1280^2 pixels and an initial kernel size $p_r = 60$ pixels (suggested KDEEB value for this resolution [vdZCT16]), and examine next the effects of the route awareness parameter p_{ra}.

For this, we generate bundles for all possible values $p_{ra} \in \{0,1,2,3\}$ for the first dataset. Figure 6(a) shows the results for $p_{ra} = 0$, *i.e.*, no route awareness. This is equivalent to using KDEEB – compare this image with Fig. 9(c) in [HET12]. The emerging bundles are indeed not affected by the road network. Setting $p_{ra} = 1$ preserves level-1 (red) routes: We see how bundles close to the red square in Fig. 6(b) follow this structure, while bundle further away follow the unorganized structure at the same locations in Fig. 6(a). Increasing p_{ra} to 2, respectively 3, increases the route awareness of bundles (Figs. 6(c,d)).

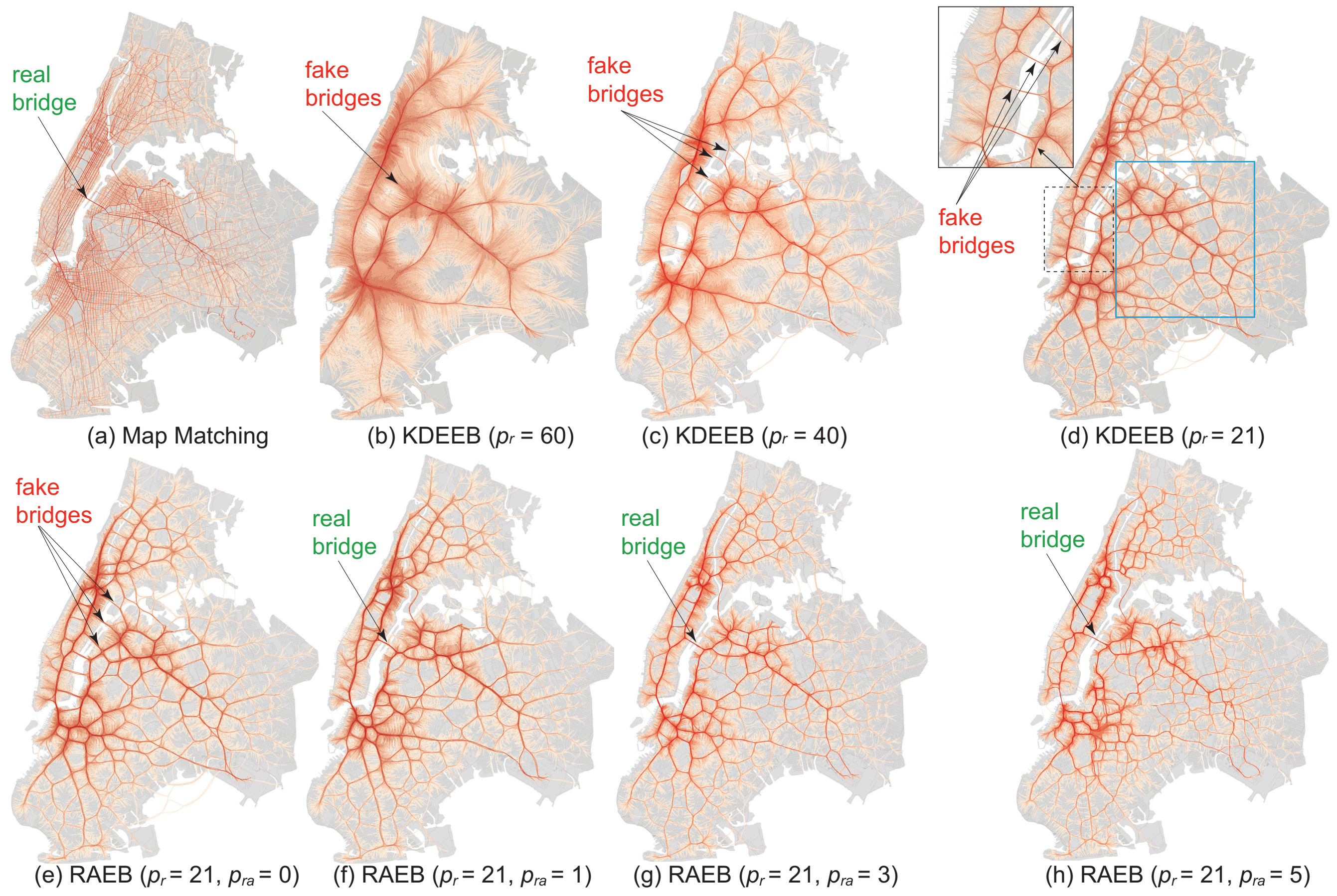


Figure 8: *Density maps of NYC taxi trips: (a) shortest paths mapped onto the road network; (b-d) KDEEB bundles with three kernel sizes; (e-h) our RAEB method with p_r = 21 and three different hierarchy levels. See Sec. 7.2.*

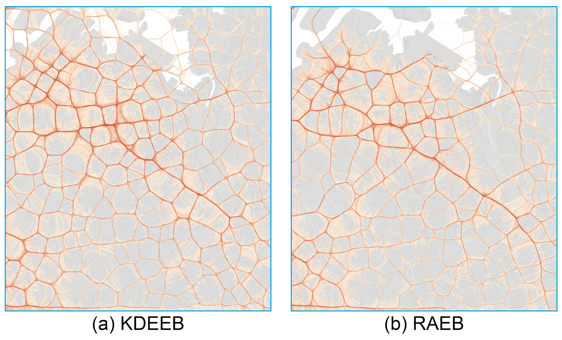


Figure 9: *Fine scale density maps of NYC taxi trips in Queens zone generated by (a) KDEEB, and (b) RAEB. See Sec. 7.2.*

In the second experiment, we use a more complex road network hierarchy (Fig. 7(left)). We set now $p_{ra} = 2$ to preserve both the red and blue routes. The resulting bundles indeed follow these structures (Fig. 7(right)). Putting it simply, RAEB allows one to trace visually end-to-end routes more easily, as these follow a given road network, as compared to the case one would use classical KDEEB.

7.2. Real Data 1: New York Taxi Trips

We next compare RAEB with KDEEB on a real-world dataset. The OD trails used here are 100K taxi trips extracted from one-month trip records, from which we retain only the pick-up (origin) and drop-off (destination) locations. The road network is extracted from OpenStreetMap (OSM) [OSM] over the Manhattan, Brooklyn, Queens, and Bronx zones in New York, where the most origins and destinations of the recorded trails are located. The raw network G has 133K nodes and 166K edges. The simplified network G' has 97K routes. We map these to the road network using a shortest path algorithm (Sec. 4.2). Figure 8(a) shows the resulting density map.

Figure 8(b) shows the bundling done by KDEEB at an image resolution of 1080^2 pixels, with the KDEEB recommended initial kernel size of 5% of the image size ($p_r = 60$ pixels) and $p_n = 10$ iterations. Several main bundles are visible. Yet, many of these do not align in any way with the road network (see *e.g.* the 'fake bridges' inset). We hypothesize that this is due to the inappropriate setting of p_r. To test this, we next set p_r with different values (40, 21), the latest being the one derived from our automatic kernel-radius-setting procedure. Figures 8(c,d) show the corresponding KDEEB bundlings. We see now finer-grained bundles, which are somehow better aligned with the main arterial roads in New York (visible as dense regions in Fig. 8(a)). Still, even with this kernel-radius variation, several bundles do not follow any road, see *e.g* the inset in Fig. 8(d) that focuses on the region between Manhattan and Brooklyn. Here, KDEEB creates bundles that actually suggest inexistent bridges. The same happens for $p_r = 40$ (Fig. 8c). Figure 8(e-h) shows the results generated by RAEB, with $p_r = 21$ pixels and a route awareness $p_{ra} \in \{0, 1, 3, 5\}$ respectively.

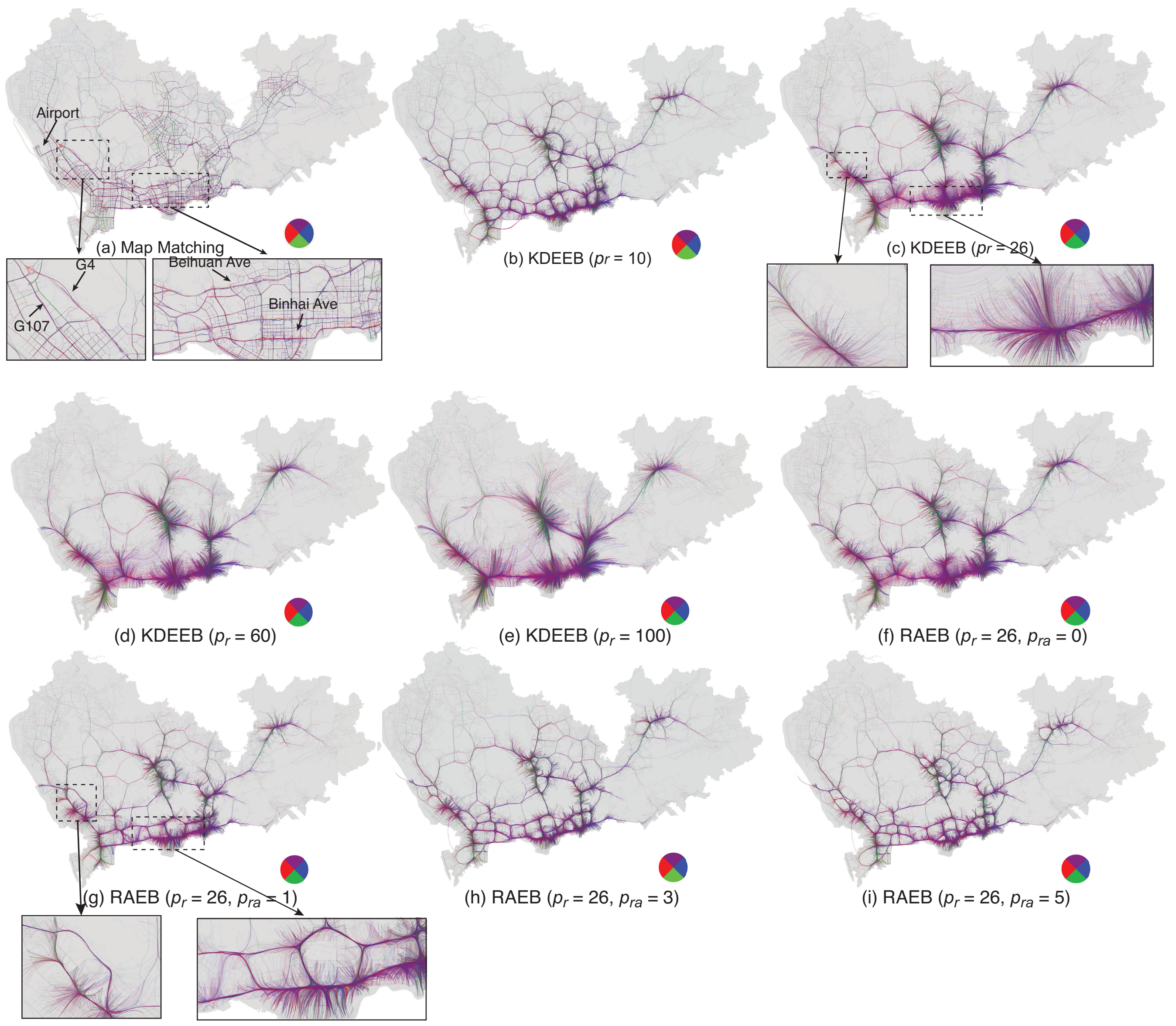

Airport
(a) Map Matching
G4
G107
Beihuan Ave
Binhai Ave
(b) KDEEB (pr = 10)
(c) KDEEB (pr = 26)
(d) KDEEB (p_r = 60)
(e) KDEEB (p_r = 100)
(f) RAEB (p_r = 26, p_{ra} = 0)
(g) RAEB (p_r = 26, p_{ra} = 1)
(h) RAEB (p_r = 26, p_{ra} = 3)
(i) RAEB (p_r = 26, p_{ra} = 5)

Figure 10: *Density maps of Shenzhen taxi trips: (a) Raw GPS records are mapped onto road network. KDEEB bundles trips on close arterial roads together (b), while RAEB keeps trips along these roads separate (c). Trails are colored according to the OD directions. See Sec. 7.3.*

The fine-grained structure produced by KDEEB is retained, but the undesirable bundle suggesting inexistent bridges are removed in all situations, except for the extreme simplification $p_{ra} = 0$.

Visualization of urban traffic should support multi-scale exploration. In our context, a good bundling method should provide *smooth transitions* when zooming in or out to see regions at different scales. To assess this, we zoom into the Queens zone (blue region in Fig. 8(d)), and re-apply KDEEB and RAEB respectively on the subset of trails in this region. Figures 9(a) and (b) show the results. Both methods generate more, finer-grained, bundles, akin to zooming in on maps. However, matching the KDEEB fine-grained bundles (Fig. 9(a)) with their coarse-scale counterparts (Fig. 8(c)) is hard, since KDEEB uses different kernel sizes p_r for the two images, because p_r depends solely on the image resolution. In contrast, RAEB generates fine-grained bundles (Fig. 9(b)) which are easier to map to the coarse-grained ones (Fig. 8(d)), as both sets are constrained by the *same* road network. Hence, RAEB creates smoother transitions between different levels of details (scales) than KDEEB, allowing one to better preserve the mental map during exploration.

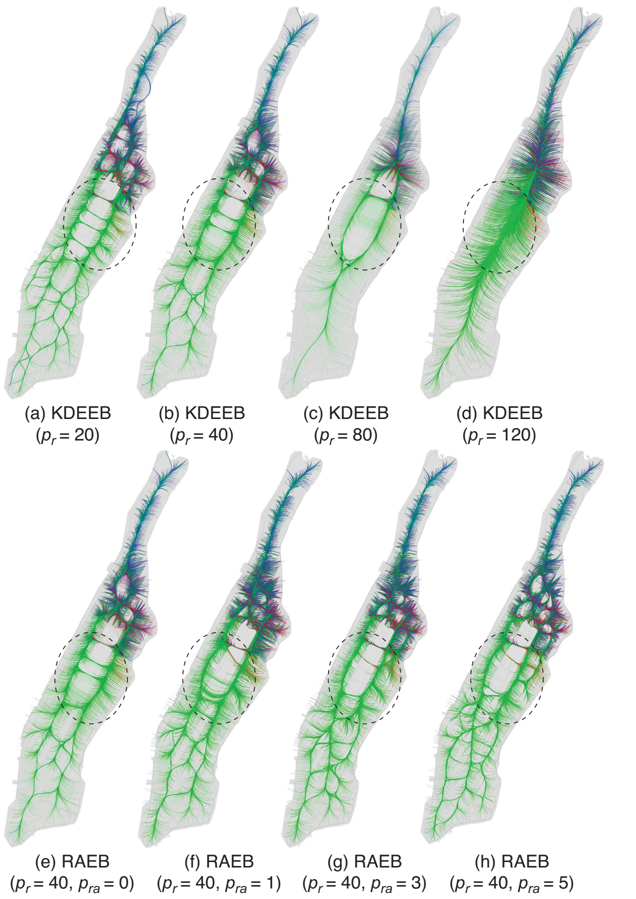
7.3. Real Data 2: Shenzhen Taxi Trips

We further evaluate RAEB using a dataset of taxi trips in Shenzhen, China. The raw road network, extracted from OSM, has 161K nodes and 177K edges, further simplified into 51K routes. Trails are one-week taxi trip records, 50K in total. Unlike the OD-only New York taxi trails, this dataset contains a sequence of GPS positions per trails, recorded every 20 seconds. Moreover, the Shenzhen dataset contains a structure of 'hierarchical' roads (a few key arterial roads spanning several secondary roads), whereas the New York road structure is much more grid-like. Finally, the Shenzhen dataset contains a quite different spatial distribution of trails.

We map these trails to the road network using the ST-matching algorithm (following Sec. 4.2). Figure 10(a) shows a density map of the resulting trips at 2160×1280 resolution, with trails color coded on direction. We already see that most taxi trips are in the more developed southern region of Shenzhen, which borders on Hong Kong. The highly unevenly-distributed trails make KDEEB's initial kernel estimation quite unreliable: Indeed, this yields $p_r = 100$ pixels (again, 5% of the drawing size), which creates too highly simplified bundles (Fig. 10(e)). Moreover, trails seem to snap in shortest-path distance to the bundle cores. Clearly, this does not obey the road network structure visible in Fig. 10(a). In contrast, our automatic kernel size estimation sets $p_r = 26$. We first usder KDEEB with this kernel size (Fig. 10(c)) and two other kernel sizes (Figs. 10(b,d)). As visible, the kernel-size change does not make KDEEB deliver significantly different bundling results. We next use our RAEB, with its recommended kernel size $(p_r = 26)$ and four different hierarchy levels $(p_{ra} \in \{0,1,3,5\})$. The coarsest simplification level, $p_{ra} = 0$, yields the same results to KDEEB. The finer simplification levels $p_{ra} \in \{1,3,5\}$ are bring increasingly more details to the view: In sparsely covered areas (north and north-east on the map), KDEEB and RAEB yield quite similar results. This is indeed expected, since in these areas there are (1) fewer roads and (2) fewer trails that follow these (main) roads. In contrast, in dense areas (south of the map), we see important differences between KDEEB and RAEB: The raw density map (Fig. 10(a), insets) tells us that expressways G4 and G107 lead traffic to Shenzhen airport (left inset), while Beihuan and Binhai avenues holds main traffic in the right inset. KDEEB wrongly merges traffic on G4 and G107 and on Beihuan and Binhai, respectively (Fig. 10(c), insets). In contrast, RAEB clearly shows these four main avenues, but simplifies traffic on smaller roads around them (Fig. 10(g), insets). Note that similar results to the simplification level $p_{ra} = 1$ (Fig. 10(g)) are obtained by the simplification levels $p_{ra} \in \{3,5\}$ (Figs. 10(h,i), respectively).

7.4. Real Data 3: Manhattan Taxi Trips

Finally, we revisit the Manhattan taxi trips dataset already illustrated in Fig. 2. Figure 11 shows the results obtained by KDEEB, for four different kernel radius values p_r, and the results obtained by our method RAEB, for the optimal kernel value $p_r = 40$ and four hierarchy levels $p_{ra} \in \{0,1,3,5\}$. Similarly to the previous real-world datasets examined in Secs. 7.2 and 7.3, we can draw several conclusions from this dataset: First, we see the extreme sensitivity of KDEEB's results as function of its kernel radius p_r: The images (a-d) in Fig. 11 suggest, depending on p_r, that there are four bridges between the west and east map side (image (a)), three bridges (image (b)), one bridge (image (c)), or no actual bridges (image (d)) – see the dashed markings in the figures. Comparing these results with the ground truth (Fig. 1(a)) shows that there should be three or four main bridges. Hence, KDEEB fails to convey this insight properly, depending on how we set p_r. Figure 11(e-h) show the results of RAEB, for the optimal kernel size $p_r = 40$ derived by our method and four different hierarchy (simplification) levels p_{ra}. The coarsest level $p_{ra} = 0$ (image (e)) is practically identical to the KDEEB bundling with the optimal kernel radius (image (b)). The other three simplification levels (images (f-h)) are quite stable, *i.e.*, show quite similar results. More importantly, these results are more similar with the ground truth (see again Fig. 1(a)) than the KDEEB results.


(a) KDEEB (p_r = 20)
(b) KDEEB (p_r = 40)
(c) KDEEB (p_r = 80)
(d) KDEEB (p_r = 120)
(e) RAEB (p_r = 40, p_{ra} = 0)
(f) RAEB (p_r = 40, p_{ra} = 1)
(g) RAEB (p_r = 40, p_{ra} = 3)
(h) RAEB (p_r = 40, p_{ra} = 5)
Figure 11: *Manhattan taxi maps generated with KDEEB, for four kernel radius p_r (a-d)); and with RAEB, using the optimal kernel size $p_r = 40$, and four simplification levels p_{ra} (e-h). See Sec. 7.4.*

8. Discussion

We next discuss RAEB's parameter settings (Sec. 8.1) and compare its performance and quality with KDEEB (Sec. 8.2). Finally, we discuss the applicability and limitations of our method (Sec 8.3).

8.1. Parameter Setting

We consider RAEB's parameter settings and compare these with KDEEB, where applicable:

Route awareness (p_{ra}): This parameter captures the key difference between RAEB and all other bundling methods we are aware of (KDEEB being just one example). When simplifying a trail set, such methods cannot control the simplification *accuracy*, since bundles are not constrained to follow an underlying (road) network. The route awareness parameter p_{ra} controls how bundles follow roads at a user-selected hierarchy level, thus trading off bundle accuracy *vs* simplification, as shown in Sec. 7.1. Setting p_{ra} is simple: A value $p_{ra} = 0$ yields results identical to KDEEB; a value $p_{ra} = N$, where N is the number of extracted road hierarchies, preserves most road details, and simplifies least. In between values trade off simplification for accuracy. A good preset we found is $p_{ra} = 1$, which corresponds to the 5% top-ranked routes following our combined criteria of length, flow, and hierarchy levels (Sec. 4.3).

Kernel size (p_r): Similar to KDE-type methods, p_r controls the degree of spatial simplification. KDE-type methods use a purely *image*

Dataset	Edge samples	p_n	Time (sec.)		Deviation (pixels)	
			KDEEB	RAEB	KDEEB	RAEB
Synthetic	4.6M	13	40.3	50.7	18.37	12.58
New York	3.1M	11	34.3	42.9	15.40	9.88
Shenzhen	1.3M	8	13.8	22.8	13.71	10.53

Table 2: *Speed and deformation comparison for KDEEB and RAEB.*

based setting of p_r, which, as we have seen, strongly oversimplifies trail sets. Our improvement here is to (automatically) set p_r based on *both* the road network geometry and its resolution in image space. This still sets p_r automatically (which is easy to use), but preserves detail much better than the p_r setting proposed by KDEEB, especially for highly non-uniform road and/or trail densities (see examples in Secs. 7.2 and 7.3).

Bundling stability p_s: KDEEB stops bundling after exceeding a user-given number of iterations p_n. Choosing this value is delicate: Too low values do not converge; too high values waste computational effort. We remove the need for this parameter by measuring the similarity of two consecutive bundling images via their normalized mutual information (*NMI*), and automatically stop bundling when this similarity exceeds a given threshold p_s. We verified that bundling visually converges for a preset $p_s = 0.8$ for all datasets and parameter combinations we worked with.

8.2. Performance and quality

Table 2 compares the computational performance of KDEEB and RAEB for the three datasets used in our studies. Both methods, implemented in Java, run on a MacBook Pro 3.1 GHz Core i7 with a Radeon Pro 560 graphics card. For comparison fairness, we used the same kernel sizes p_r and total bundling iterations p_n (both computed by our proposed approach) for both methods.

In absolute terms, we see that bundling time is mainly influenced by the number of trail samples and kernel size, in line with earlier studies [HET12, vdZCT16]. The running times for KDEEB are comparable with those in the original work [HET12]. Relatively speaking, RAEB is about 20% slower than KDEEB. This is because RAEB must also evaluate bundling stability to check for termination (Sec. 6). This operation is linear in image size and number of bundling steps and, for the considered datasets, takes about 1 second/iteration. If we subtract this time from RAEB's total, we see that RAEB is actually *faster* than KDEEB. Indeed, this is because our estimated kernel size (p_r) is in general lower than KDEEB's, so we compute the density map faster than KDEEB. Finally, we note that RAEB can be easily and significantly accelerated using GPU parallelization, precisely as done earlier for KDEEB [vdZCT16].

Table 2 also compares the deviation between the (KDEEB and RAEB) bundles and the input OD trails (see Sec. 6.2). We see that RAEB is closer to the original trails than KDEEB – roughly 30% for the synthetic and the New York taxi datasets. The deviation difference is smaller for the Shenzhen dataset, mainly because most trails in this dataset are concentrated in a smaller region of the map (Fig. 10). Still, as the same figure shows, RAEB's smaller deviations are significant in interpreting the original trails more correctly in these regions. Overall, we conclude that RAEB deforms the raw input data markedly less than KDEEB, which is good for preserving the meaning of the spatial trails.

8.3. Applicability and Limitations

RAEB can be readily extended to any scenario where one has (a) a set of 2D spatial trails, defined as straight lines or as polylines; and (b) a polyline-like graph-drawing that defines a skeletal structure that the trails should obey as well as possible. Hence, RAEB directly applies to other types of map-related traffic data, *e.g.* commuter trips by public transportation, or people traces derived from phone call records. We also note that trails do not *have* to be defined as following the network: Indeed, if desired, one can use an arbitrary graph drawing to constrain a trail set, as long as this constraining makes sense for the problem at hand. Separately, RAEB's added-value is as good as the underlying data it uses: If route importance is wrongly estimated (Sec. 4.3), then the simplified RAEB bundling will be potentially misleading. If only OD data is available, then one will not be able to accurately reason about actual geographic trails. Yet, these are generals problem related to the trustworthiness and/or completeness of *data* underlying any visualization method, not specific to RAEB.

By construction, RAEB proposes a *trade-off* between simplification and trail accuracy preservation, via the road awareness parameter p_{ra}. At one extreme, RAEB can preserve all trail road-related geometry, but this leads to no or negligible simplification (much like vector maps). At the other extreme, RAEB can completely ignore the road network, yielding strong simplification but potentially too large deformations (much like KDE-type methods). To our knowledge, RAEB is the first method that explicitly offers this trade-off for generic OD trail sets. Which actual trade-off between simplification and trail preservation is optimal strongly depends on the application type. For instance, in scenarios requiring precise positions, such as querying movements passing through waypoints [KTW*13] or road segment [WCW*14], visual hints would be required to remind the user on how much bundling deforms the actual data. However, the focus of our work here is not to propose such an optimal trade-off, but the *technique* that allows one to actually specify a trade-off value and obtain bundled results that obey this value.

A limitation of our current approach regards *directional* trails. Our current RAEB implementation does not consider trail directions during bundling, for implementation simplicity, and since we wanted to compare our results with the original KDEEB which also does not consider directions. However, direction is of recognized importance for studying OD trails in urban traffic [ZFMA*16]. RAEB can be easily adapted to perform directional bundling in precisely the same way as it was done for KDEEB by the CUBu method [vdZCT16], or, alternatively, by considering the more powerful but more complex directional bundling adaption of KDEEB proposed by the FFTEB method [LHT17a].

A separate limitation is that the route awareness parameter (p_{ra}) works in a global way. On the one hand, this makes our method very simple, as the user only needs to set a single value to simplify an entire dataset. On the other hand, this lacks local control. To have the latter, we could imagine setting p_{ra} locally, by *e.g.* specifying which geographic areas are to be simplified more and which are to be kept at a high detail level. Incorporating this in the hierarchical road network construction (Sec. 4.3) is quite simple; the challenge is to find an easy to use but effective user interface to allow specifying p_{ra} locally. As such, we leave this extension, and its evaluation by user studies, for future work.

Finally, an important issue we do not yet tackle is showing the *local* *errors* introduced by bundling. By definition, bundling deforms trails to yield a simplified, albeit inaccurate, view of a trail-set. Showing how much, and where, bundled trails differ from original ones (in our case, the road network) is an open question [vdZCT16]. Computing such errors is simple; showing them is however tricky, since we already use visual variables such as color for direction, and opacity for trail density, respectively. Yet, our quantitative evaluation shows that we create far smaller deformations than state-of-the-art bundling methods (Tab. 2).

9. Conclusion and future work

In this paper, we have presented Route-Aware Edge Bundling (RAEB), a novel method for creating simplified visualizations of urban trail sets. RAEB adapts Kernel Density Estimation (KDE)-type bundling methods to alleviate several of their known problems which make them hard to use for urban trail data, as follows. First and foremost, we constrain trails to a given road network, by offering the user a simple way to trade off the degree of simplification (bundling) *vs* the degree of respecting the hierarchically modeled road network. Secondly, we present a more involved heuristic of controlling the kernel size, a central parameter in KDE bundling, and show how our method yields better detail preservation and also more stable bundling results *vs* the image resolution, which in turn helps multi-scale visual exploration by preserving the user's mental map. Thirdly, we propose a new method for determining bundling convergence based on the convergence of the visual result rather than user parameters which are not always easy to set. We compared our method with KDEEB, a state-of-the-art method in the KDE class, on three datasets, including two large real-world urban trail sets. Qualitatively, we showed how RAEB yields bundles that respect the original trail sets and underlying road networks visibly better than KDEEB, while still achieving visual simplification. Quantitatively, we showed that RAEB has comparable running times as KDEEB, while it achieves lower overall deformations of the input trail sets.

RAEB opens several directions for future research, as follows. First, its main technical contributions (constraining bundling to a given network-like drawing, trading off trail position preservation *vs* simplification, and automating termination) can be easily incorporated into most other bundling methods, *e.g.* SBEB [EHP*11] and FDEB [HvW09]. Doing so would be interesting, as such methods in turn offer different bundling styles and bundling control parameters than KDEEB. Secondly, accelerating RAEB using GPU parallelization is a very low, and interesting, hanging fruit, following similar developments for KDEEB [vdZCT16]. Thirdly, we plan to adapt and apply RAEB to other application areas where spatially-constrained bundling is required, *e.g.*, brain network simplified visualization [BSL*14, YSD*17]. Last but not least, our proposal for measuring the bundling faithfulness by the Fréchet distance between the output bundles and the ground truth road network along which input trails should go, can be refined by proposing richer distance metrics that can model more than local Euclidean distances. Such metrics can incorporate *e.g.* local bundle orientation and curvature, thereby capturing more involved priors known about trail sets in specific applications.

Acknowledgment. The authors wish to thank anonymous reviewers for their constructive comments. The work was supported in part by National Natural Science Foundation of China (61802388), the CAS grant (GJHZ1862), Shenzhen Science and Technology Program (JCYJ20180507182222355).

References

[AA11] ANDRIENKO N., ANDRIENKO G.: Spatial generalization and aggregation of massive movement data. *IEEE TVCG 17*, 2 (2011), 205–219. 1

[AAC*17] ANDRIENKO G., ANDRIENKO N., CHEN W., MACIEJEWSKI R., ZHAO Y.: Visual analytics of mobility and transportation: State of the art and further research directions. *IEEE T-ITS 18*, 8 (2017), 2232–2249. 2

[BPSW05] BRAKATSOULAS S., PFOSER D., SALAS R., WENK C.: On map-matching vehicle tracking data. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)* (2005), pp. 853–864. 2

[BRH*17] BACH B., RICHE N. H., HURTER C., MARRIOTT K., DWYER T.: Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE TVCG 23*, 1 (2017), 541–550. 3

[BSL*14] BÖTTGER J., SCHÄFER A., LOHMANN G., VILLRINGER A., MARGULIES D. S.: Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE TVCG 20*, 3 (2014), 471–480. 3, 12

[CGW15] CHEN W., GUO F., WANG F.-Y.: A survey of traffic data visualization. *IEEE T-ITS 16*, 6 (2015), 2970–2984. 2

[CKS*16] CORNEL D., KONEV A., SADRANSKY B., HORVÁTH Z., BRAMBILLA A., VIOLA I., WASER J.: Composite flow maps. *Comput. Graph. Forum 35*, 3 (2016), 461–470. 2

[CZQ*08] CUI W., ZHOU H., QU H., WONG P. C., LI X.: Geometry-based edge clustering for graph visualization. *IEEE TVCG 14*, 6 (2008), 1277–1284. 1, 3

[DWL08] DODGE S., WEIBEL R., LAUTENSCHÜTZ A.-K.: Towards a taxonomy of movement patterns. *Info. Vis. 7*, 3 (2008), 240–252. 1

[EHP*11] ERSOY O., HURTER C., PAULOVICH F., CANTAREIRO G., TELEA A.: Skeleton-based edge bundling for graph visualization. *IEEE TVCG 17*, 12 (2011), 2364–2373. 3, 12

[EM94] EITER T., MANNILA H.: *Computing Discrete Frechet Distance*. Technical Report CDTR 94/64, Christian Doppler Laboratory for Expert Systems. TU Vienna, Austria, 1994. 6

[FKSS13] FERREIRA N., KLOSOWSKI J. T., SCHEIDEGGER C. E., SILVA C. T.: Vector field k-means: Clustering trajectories by fitting multiple vector fields. *Computer Graphics Forum 32*, 3 (2013), 201–210. 2

[FPV*13] FERREIRA N., POCO J., VO H. T., FREIRE J., SILVA C. T.: Visual exploration of big spatio-temporal urban data: A study of New York city taxi trips. *IEEE TVCG 19*, 12 (2013), 2149–2158. 2, 4

[GLW11] GUDMUNDSSON J., LAUBE P., WOLLE T.: Computational movement analysis. In *Handbook of Geographic Information*. Springer, 2011, pp. 423–438. 6

[GZ14] GUO D., ZHU X.: Origin-destination flow data smoothing and mapping. *IEEE TVCG 20*, 12 (2014), 2043–2052. 1, 2

[HET12] HURTER C., ERSOY O., TELEA A.: Graph bundling by kernel density estimation. *Comput. Graph. Forum 31*, 3pt1 (2012), 865–874. 1, 3, 4, 6, 7, 11

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG 12*, 5 (2006), 741–748. 1, 3

[HPNT18] HURTER C., PUECHMOREL S., NICOL F., TELEA A.: Functional decomposition for bundled simplification of trail sets. *IEEE TVCG 24*, 1 (2018), 500–510. 3

[HvW09] HOLTEN D., VAN WIJK J. J.: Force-directed edge bundling for graph visualization. *Comput. Graph. Forum 28*, 3 (2009), 983–990. 1, 3, 5, 12

[KJW*18] KIM S., JEONG S., WOO I., JANG Y., MACIEJEWSKI R., EBERT D. S.: Data flow analysis and visualization for spatiotemporal statistical data without trajectory information. *IEEE TVCG 24*, 3 (2018), 1287–1300. 2

[KSBE18] KRÜGER R., SIMEONOV G., BECK F., ERTL T.: Visual interactive map matching. *IEEE TVCG 24*, 6 (2018), 1881 – 1892. 2, 5

[KTW*13] Krüger R., Thom D., Wörner M., Bosch H., Ertl T.: TrajectoryLenses – a set-based filtering and exploration technique for long-term trajectory data. *Comput. Graph. Forum 32*, 3pt4 (2013), 451–460. 2, 4, 11

[LBA10a] Lambert A., Bourqui R., Auber D.: 3D edge bundling for geographical data visualization. In *Proc. Intl. Conf. Info. Vis.* (2010), pp. 329–335. 3

[LBA10b] Lambert A., Bourqui R., Auber D.: Winding roads: Routing edges into bundles. *Comput. Graph. Forum 29*, 3 (2010), 853–862. 3

[LHT17a] Lhuillier A., Hurter C., Telea A.: FFTEB: Edge bundling of huge graphs by the fast fourier transform. In *Proc. IEEE PacificVis* (2017), pp. 190–199. 1, 3, 11

[LHT17b] Lhuillier A., Hurter C., Telea A.: State of the art in edge and trail bundling techniques. *Comput. Graph. Forum 36*, 3 (2017), 619–645. 1, 2, 3

[LHW07] Lee J.-G., Han J., Whang K.-Y.: Trajectory clustering: a partition-and-group framework. In *Proc. ACM SIGMOD* (2007), pp. 593–604. 6

[LLCM12] Luo S.-J., Liu C.-L., Chen B.-Y., Ma K.-L.: Ambiguity-free edge-bundling for interactive graph visualization. *IEEE TVCG 18*, 5 (2012), 810–821. 3

[LZZ*09] Lou Y., Zhang C., Zheng Y., Xie X., Wang W., Huang Y.: Map-matching for low-sampling-rate GPS trajectories. In *Proc. ACM SIGSPATIAL* (2009), pp. 352–361. 4

[MCV*97] Maes F., Collignon A., Vandermeulen D., Marchal G., Suetens P.: Multimodality image registration by maximization of mutual information. *IEEE Trans. Med. Imaging 16*, 2 (1997), 187–198. 6

[NYC] NYC Taxi and Limousine Commission: TLC Trip Record Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. 4

[OSM] OSMF: Open Street Map. https://www.openstreetmap.org. 5, 8

[PHT15] Peysakhovich V., Hurter C., Telea A.: Attribute-driven edge bundling for general graphs with applications in trail analysis. In *Proc. IEEE PacificVis* (2015). 3

[PXYH05] Phan D., Xiao L., Yeh R., Hanrahan P.: Flow map layout. In *Proc. IEEE InfoVis* (2005), pp. 219–224. 1, 2

[QON07] Quddus M. A., Ochieng W. Y., Noland R. B.: Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies 15*, 5 (2007), 312–328. 2

[Qud06] Quddus M. A.: *High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications*. Phd thesis, Imperial College London, 2006. 2

[SHH11] Selassie D., Heller B., Heer J.: Divided edge bundling for directional network data. *IEEE TVCG 17*, 12 (2011), 2354–2363. 1, 3

[SHvdW*16] Scheepens R., Hurter C., van de Wetering H., , van Wijk J. J.: Visualization, selection, and analysis of traffic flows. *IEEE TVCG 22*, 1 (2016), 379–388. 1

[Som14] Sommer C.: Shortest-path queries in static networks. *ACM Comput. Surv. 46*, 4 (2014), 1–31. 5

[SWvdW*12] Scheepens R., Willems N., van de Wetering H., , van Wijk J. J.: Interactive density maps for moving objects. *IEEE CG&A 32*, 1 (2012), 56–66. 1

[TP15] Thöny M., Pajarola R.: Vector map constrained path bundling in 3D environments. In *Proc. ACM SIGSPATIAL Intl. Workshop on GeoStreaming* (2015), pp. 33–42. 3

[VBS11] Verbeek K., Buchin K., Speckmann B.: Flow map layout via spiral trees. *IEEE TVCG 17*, 12 (2011), 2536–2544. 1

[vdZCT16] van der Zwan M., Codreanu V., Telea A.: CUBu: Universal real-time bundling for large graphs. *IEEE TVCG 22*, 12 (2016), 2550–2563. 3, 4, 5, 7, 11, 12

[vLBR*16] von Landesberger T., Brodkorb F., Roskosch P., Andrienko N., Andrienko G., Kerren A.: MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE TVCG 22*, 1 (2016), 11–20. 2

[WCW*14] Wang F., Chen W., Wu F., Zhao Y., Hong H., Gu T., Wang L., Liang R., Bao H.: A visual reasoning approach for data-driven transport assessment on urban road. In *Proc. IEEE VAST* (2014), pp. 103–112. 4, 5, 11

[WDS10] Wood J., Dykes J., Slingsby A.: Visualisation of origins, destinations and flows with od maps. *Cartogr. J. 47*, 2 (2010), 117–129. 2

[WF09] Wilkinson L., Friendly M.: The history of the cluster heat map. *The American Statistician 63*, 2 (2009), 179–184. 2

[WHB*12] Wang P., Hunter T., Bayen A. M., Schechtner K., González M. C.: Understanding road usage patterns in urban areas. *Scientific Reports 2* (2012), 1001. 1

[WvdWvW09] Willems N., van de Wetering H., van Wijk J. J.: Visualization of vessel movements. In *Proc. EuroVis* (2009), pp. 959–966. 2

[YSD*17] Yang X., Shi L., Daianu M., Tong H., Liu Q., Thompson P.: Blockwise human brain network visual comparison using nodetrix representation. *IEEE TVCG 23*, 1 (2017), 181 – 190. 3, 12

[ZFMA*16] Zeng W., Fu C. W., Müller Arisona S., Erath A., Qu H.: Visualizing waypoints-constrained origin-destination patterns for massive transportation data. *Comput. Graph. Forum 35*, 8 (2016), 95–107. 2, 4, 11

[ZXYQ13] Zhou H., Xu P., Yuan X., Qu H.: Edge bundling in information visualization. *Tsinghua Sci. Technol. 18*, 2 (2013), 145–156. 3

[ZYC*08] Zhou H., Yuan X., Cui W., Qu H., Chen B.: Energy-based hierarchical edge clustering of graphs. In *Proc. IEEE PacificVis* (2008), pp. 55–61. 3